The Pragmatic Programmers

TURING 图灵程序设计丛书

HTML5 and CSS3

Develop with Tomorrow's Standards Today

HTML5和CSS3实例教程

[美] Brian P. Hogan 著

李杰 刘晓娜 朱嵬 译

柳靖 审校



人民邮电出版社

POSTS & TELECOM PRESS

数字版权声明

Brian P. Hogan

1995年起便开始以自由职业者的身份开发专业网站并提供咨询服务，目前常使用Ruby、jQuery、HTML5和CSS3构建Web应用。他乐于讲述并撰写与Web设计和开发有关的内容，倡导为残障人士（特别是视觉障碍者）研发辅助功能。

图书在版编目（ＣＩＰ）数据

HTML5和CSS3实例教程 / (美) 霍根 (Hogan,B.P.)
著 ; 李杰, 刘晓娜, 朱嵬译. -- 北京 : 人民邮电出版社,
2012.1
（图灵程序设计丛书）
书名原文: HTML5 and CSS3:Develop with
Tomorrow's Standards Today
ISBN 978-7-115-26724-5

Ⅰ. ①H… Ⅱ. ①霍… ②李… ③刘… ④朱… Ⅲ. ①
超文本标记语言，HTML 5－程序设计－教材②网页制作工
具，CSS 3－教材 Ⅳ. ①TP312②TP393.092

中国版本图书馆CIP数据核字(2011)第231332号

内 容 提 要

本书共分 3 部分，集中讨论了 HTML5 和 CSS3 规范及其技术的使用方法。首先是规范概述，介绍了新的结构化标签、表单域及其功能（包括自动聚焦功能和占位文本）和 CSS3 的新选择器。接下来是 HTML 对视频和音频的支持，讲述了画布上的图形绘制及 CSS 阴影、渐变和变换的使用方法。最后介绍使用 HTML5 的客户端特性（包括 Web Storage、Web SQL Databases 以及离线支持）建立客户端应用，使用 HTML5 实现跨域消息和数据传送，以及操作浏览器历史等的方法。

本书适合所有使用 HTML 和 CSS 的 Web 开发人员学习参考。

图灵程序设计丛书

HTML5和CSS3实例教程

◆ 著　[美] Brian P. Hogan
译　李　杰　刘晓娜　朱　嵬
审　校　柳　靖
责任编辑　卢秀丽
执行编辑　毛倩倩

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷

◆ 开本：800×1000　1/16
印张：13
字数：288千字　2012年 1 月第 1 版
印数：1－4 000册　2012年 1 月北京第 1 次印刷

著作权合同登记号　图字：01-2011-6291 号

ISBN 978-7-115-26724-5

定价：39.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223

反盗版热线：(010)67171154

版 权 声 明

目　　录

第三部分　HTML5 延伸

第 1 章

HTML5 和 CSS3 概述

HTML5[1]和 CSS3[2]并非只是 W3C（万维网联盟）及其下辖工作组提出的两项新标准。它们分别代表了下一代的 HTML 和 CSS 技术，开发人员可以在日常工作中使用它们来更好地构建时新的 Web 应用。在深入探讨 HTML5 和 CSS3 之前，我们先讨论一下 HTML5 和 CSS3 的优势，以及应用过程中需要面对的难点。

1.1 一个新的 Web 开发平台

HTML 中的众多新功能都围绕着一个核心目标，即构建一套更强大的 Web 应用开发平台。从更多的描述性标签、更好的跨站及跨窗口通信到动画及更强的多媒体支持，HTML5 开发人员拥有大量新工具实现更好的用户体验。

1.1.1 更多的描述性标记

HTML 的每个版本都会引入一些新标记，但之前没有哪个版本能像 HTML5 这样，引入如此多的直接用于描述内容的标记。在第 2 章，你将看到用于定义头部（header）、尾部（footer）、导航区段、侧边栏和正文的元素。此外，你还将了解到数值范围的表示、进度条的生成，以及如何使用可定制的数据属性来对数据进行标记。

1.1.2 较少依赖于插件的多媒体支持

现在，播放视频、音频以及浏览矢量图形可以不必使用 Flash 或 Silverlight 了。尽管基于 Flash 的视频播放器简单易用，但苹果公司的移动设备并不支持 Flash。考虑到这一块市场的重要性，在设备不支持 Flash 时，你需要为视频播放提供替代方案。在第 7 章，我们将讨论如何通过 HTML5 的 `audio` 标签和 `video` 标签实现有效应变。

---

① 要查看 HTML5 规范可以访问 http://www.w3.org/TR/html5/。

② CSS3 规范分散在多个不同规范的模块中，访问 http://www.w3.org/Style/CSS/current-work 可以查看其最新进展。

1.1.3 更强大的 Web 应用

为了让 Web 应用内容更加丰富、交互性更强，开发人员曾绞尽脑汁，尝试了包括 ActiveX 控件和 Flash 在内的各种手段。HTML5 提供了一些令人称奇的功能，在某些情况下，你甚至可以完全放弃使用第三方技术。

1.1.4 跨文档消息通信

Web 浏览器会阻止不同域间的脚本交互或影响。这种限制机制能够防范跨站脚本对那些毫无戒备的站点访问者的攻击，进而达到保护终端用户的目的。

但每枚硬币都有两面，浏览器会阻止所有跨站脚本的交互，即使是我们自己写的完全可信任的脚本也不行。为了解决这一问题，HTML5 引入了一套安全且易于实现的应对方案。在 10.2 节，我们将详细讲解相关内容。

1.1.5 Web Sockets

HTML5 提供了对 Web Sockets 的支持，通过 Web Sockets，开发人员能够实现与服务器间的持久连接。如果要获取进度更新，你不必再像以往那样轮询后端服务器，取而代之的方式是用网页订阅某个套接字，当有新数据到达时，后端服务器会主动向订阅用户推送通知。我们会在 10.3 节中简单介绍相关内容。

1.1.6 客户端存储

我们倾向于将 HTML5 看做一项 Web 技术，但基于其提供的 Web Storage 和 Web SQL Database API，我们在浏览器中构建的 Web 应用能够完全在客户端持久化数据！在第 9 章，我们会展示如何使用这些 API。

1.1.7 更精美的界面

用户界面是 Web 应用的重要组成部分，为了让浏览器能够渲染出所期望的界面效果，我们每天都在极努力地工作。以前为了给表格添加样式或者绘制圆角，我们除了使用 JavaScript 库或添加大量冗余标记外别无他法。现在，HTML5 和 CSS3 的出现让以往的做法成为了历史。

1.1.8 更强大的表单

HTML5 提供了功能更为强大的用户界面控件。长期以来，我们只能使用 JavaScript 和 CSS

来构造滑块、日期选择器和颜色选择器，而在 HTML5 中，它们都被定义成了真正的元素，就像下拉列表、复选框和单选按钮一样。我们会在第 3 章详细描述如何使用它们。尽管不是每个浏览器都兼容这些新的表单控件，你仍然需要对它们保持关注，特别是在开发 Web 应用的时候。除了不依赖 JavaScript 库就能提升可用性之外，HTML5 还带来了可访问性的提升。屏幕阅读器和其他浏览程序会通过一些特殊方式实现 HTML5 中的表单控件，以方便残障人士使用。

1.1.9 提升可访问性

使用 HTML5 新元素清晰描述的内容更便于屏幕阅读器等程序使用。例如对于某网站的导航，它们更容易找到 nav 标签，而不是特定的 div 或无序列表。尾部、侧边栏等内容都能够被轻松地重新排序或整体跳过。一般的页面解析会变得更加容易，从而为那些依靠辅助技术浏览网页的人们带来更好的体验。另外，元素的新属性能够指明元素的角色，以便屏幕阅读器更容易处理这些元素。在第 5 章，你将了解到如何控制这些新属性来让现有的屏幕阅读器处理它们。

1.1.10 先进的选择器

利用 CSS3 选择器，你可以识别出表格的奇数行和偶数行、所有处于选中状态的复选框，甚至是文章中的最后一段。使用的代码和标记更少了，能完成的事情却更多了。此外，对于一些无法修改 HTML 的情况，CSS3 选择器简化了我们的样式设定过程。我们会在第 4 章探讨如何有效利用这些选择器。

1.1.11 视觉效果

文本和图像上的阴影效果会让网页更具层次感，而渐变效果可以增加页面的维度。使用 CSS3，你可以直接为元素添加阴影和渐变效果，而不需要借助背景图像或额外的标记。除了阴影和渐变效果，我们还可以使用 CSS3 中的变换（transformation）来制作圆角或旋转元素。本书第 8 章将详述以上内容。

1.2 向后兼容

今天，开发人员使用 HTML5 的最重要原因之一是它被大部分浏览器支持。现在，即使在 IE6 中，你也可使用 HTML5 来开发，逐步转换使用的标记。编写完成后的 HTML 代码甚至可以用 W3C 的验证服务来进行验证（当然，这是有条件的，因为标准也在演进中）。

如果你用过 HTML 或 XML，一定遇到过文档类型声明（doctype declaration）。它被用来通知验证器和编辑器哪些标签和属性是你可以使用的，以及文档应该以何种方式加以组织。此外，很

多 Web 浏览器会根据它来检测如何渲染网页。通常情况，有效的文档类型会使浏览器在“标准模式”下渲染网页。

比较一下，下面是许多网站用到的相当冗长的 *XHTML 1.0 Transitional* 文档类型：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

而 HTML5 文档类型声明非常简单：

Download **html5_why/index.html**

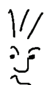
```
<!DOCTYPE html>
```

在文档顶端使用上面的这个文档类型声明后，你就可以使用 HTML5 了。

当然，你不能使用目标浏览器不支持的那些 HTML5 新元素，不过，这并不妨碍你的文档进行 HTML5 验证。

1.3 未来之路崎岖不平

HTML5 和 CSS3 的普及道路上还有许多障碍，其中有些显而易见，有些则不那么明显。



小乔爱问……

可是，我喜欢 XHTML 中的自闭合标签，在 HTML5 中我还能使用它们吗?

当然可以！许多开发人员喜欢 XHTML，因为 XHTML 对标记的使用有着更严格的要求。XHTML 文档要求属性值必须在引号中，内容标签必须自闭合，属性名称必须小写，以及只有格式良好的标记才可以发布到万维网上。改用 HTML5 后，你无需改变使用方式。若使用 HTML5 语法或 XHTML 语法，HTML5 文档仍然有效，不过，你需要理解使用自闭合标签的含义。

多数 Web 服务器在返回 HTML 网页时为其指定的是 `text/html` MIME 类型，因为 IE 无法正确处理与 XHTML 页面相关联的 MIME 类型 `application/xml+xhtml`。基于此，浏览器往往会去掉自闭合标签，因为自闭合标签在 HTML5 之前被视为无效的 HTML。例如，如果你在 `div` 前面编写了自闭合 `script` 标签，如下所示：

```
<script language="javascript" src="application.js" />
<h2>Help</h2>
```

浏览器会移去用于自闭合的斜杠，然后，渲染器会认为h2包含在永远都无法闭合的script标签内。这正是为什么编写script标签时都会对其添加一个结束的</script>标签进行显式闭合的原因，尽管自闭合标签是有效的XHTML标记。

所以，如果在 HTML5 文档中使用自闭合标签，你需要知道这些细节，因为页面会以 `text/html` MIME 类型加以渲染。如果想了解更多此方面的知识，可以访问 http://www.webdevout.net/articles/beware-of-xhtml#myths。

蛋糕与糖霜

我喜欢蛋糕。我更喜欢派，不过，蛋糕也不错。我更喜欢洒满糖霜的蛋糕。

当你开发 Web 应用时，必须时刻牢记漂亮的用户界面和绚丽的 JavaScript 效果就是洒在蛋糕上的糖霜。即便没有那些装饰，网站也可以相当不错，这就像一块蛋糕，以它为基础，你才能往上洒糖霜。

我遇到过一些不喜欢糖霜的人。他们会从蛋糕上将其刮掉。我还见过有些人，他们因为各种各样的原因使用没有 JavaScript 的 Web 应用。

为这些人烘烤一块非凡的蛋糕吧，然后再撒上糖霜。

1.3.1 IE

IE 是目前世界上应用最广的浏览器，IE8 及其以前的版本对 HTML5 和 CSS3 的支持非常弱。IE9 做了改进，但仍未广泛使用这些新技术。这并不是说我们在网站中完全不能用 HTML5 和 CSS3。我们的网站可以在 IE 中正常运行，但它并不一定要像为 Chrome 和 Firefox 开发的版本那样运行。我们可以提供备用解决方案，这样既不会惹怒用户，也不会失去客户。

1.3.2 可访问性

用户需要与网站交互，无论他们是有视力障碍或听力障碍，还是使用旧的浏览器、连接慢的网络或者移动设备上网。HTML5 引入了一些新元素，如 `audio`、`video` 和 `canvas`。音频和视频一直都有可访问性问题，而 `canvas` 又提出了新的挑战。基于 `canvas` 元素，我们能够在 HTML 文档中使用 JavaScript 创建矢量图形。这不仅给会视力障碍的用户带来问题，也会给 5%左右禁用了 JavaScript 的用户制造麻烦[①]。

正像我们要考虑使用 IE 的人群一样，在使用新技术和为 HTML5 元素提供适当备用方案时，我们也应该考虑到可访问性。

---

① 参见 http://visualrevenue.com/blog/2007/08/eu-and-us-javascript-disabled-index.html。

1.3.3 废弃的标签

HTML5 引入了许多新元素，但同时新规范也废弃了不少常见元素，你可能会在自己的页面中找到这些元素。[1]如果能做到不使用它们，那就表明你进步了。

首先是删除了几个表现性元素。如果你在代码中发现了它们，马上清除它们！用语义正确的元素代替它们，并使用 CSS 来确保渲染后的效果。

- basefont
- big
- center
- font
- s
- strike
- tt
- u

其中的一些标签含义非常模糊，但在许多用 Dreamweaver 之类的所见即所得编辑器维护的网页里，你还是能看到很多 font 和 center 标签。

除了表现元素以外，规范还移除了对框架的支持。框架一直受到像 PeopleSoft、Microsoft Outlook Web Access 这些企业级 Web 应用的推崇，它甚至被用于定制门户网站。尽管框架得到了广泛应用，但其带来了如此多的可用性和可访问性问题，以至于规范中不得不将它们移除。这意味以下元素不会再出现在规范中：

- frame
- frameset
- noframes

你应该使用 CSS 或 JavaScript 来设计页面布局，而不是使用框架。如果需要使用 frame 元素来确保应用中的每个页面都有相同的头部、尾部和导航，那么借助 Web 开发框架提供的一些工具，你同样可以完成这件事。因为有了更好的选择，如下一些元素也被移除了：

- abbr 取代了 acronym；
- object 取代了 applet；
- ul 取代了 dir。

---

① 参见 http://www.w3.org/TR/html5-diff/。

除了废弃的元素外，还有许多属性将不再有效。这包括如下的表现属性：

- align；
- body 标签上的 link、vlink、alink 和 text 属性；
- bgcolor；
- height 和 width；
- iframe 元素上的 scrolling 属性；
- valign；
- hspace 和 vspace；
- table 标签上的 cellpadding、cellspacing 和 border 属性。

如果像下面这样在链接上使用 target 属性：

```
<a href="http://www.google.com" target="_blank">
```

那么，最好通过使用 JavaScript 来代替 target 属性，因为 target 属性在规范中被废弃了。

head 标签上的 profile 属性将不再受到支持，但许多 WordPress 模板中都使用了这个属性。

最后，img 和 iframe 元素的 longdesc 属性也被移除了，这让倡导可访问性的人多少有点失望，因为 longdesc 用于向屏幕阅读器用户提供附加描述性信息，而这种方法已为人们所接受。

如果计划在现有网站中使用 HTML5，你需要找到这些元素，将其移除或用更语义化的元素来代替。记得要用 W3C 验证服务[①]来检验网页，这样有助于找到那些废弃的标签和属性。

1.3.4　企业利益的竞争

IE 并不是唯一一个在支持 HTML5 和 CSS3 上步履蹒跚的浏览器，Google、Apple 和 Mozilla 基金会都有各自的时间表，要争夺霸权。他们就视频和音频编码的支持问题争论不休，都在自己的浏览器中实现了各自的想法。例如，audio 元素在 Safari 中能够播放 MP3 音频文件，但 ogg 文件无法运行。然而，Firefox 支持 ogg 文件，却不支持 MP3 文件。

差异终会得到解决的。在此期间，我们也可以做出明智选择，要么仅支持目标用户所使用的部分类型的浏览器，要么支持全部浏览器，针对每种浏览器都给出不同的实现，直到标准最终敲定。其实，具体操作并不像听起来这么麻烦，本书第 7 章将详细介绍这个问题。

---

① 参见 http://validator.w3.org/。

1.3.5 HTML5 和 CSS3 仍在改进

刚刚谈到的并不是最终规范，这意味着这些规范中的任何东西都可能改变。目前，Firefox、Chrome、Safari 已经能够很好地支持 HTML5 了，如果规范改变，浏览器会随之改变，这可能会导致部分网站挂掉。在编写本书的过程中，CSS3 的 `box-shadow` 属性先被移除，之后又被重新添加到了规范中，Web Sockets 协议也经过了修改，完全突破了客户端与服务器之间的通信模式。

跟上 HTML5 和 CSS3 进步的步伐，不断更新自己的知识，对开发人员而言，做到这些无疑是件好事。还有一件好事，本书讨论的许多内容在今后很长一段时间内都适用。

倘若某种目标浏览器无法正常运行 HTML5 代码，可以使用 JavaScript 和 Flash 实现来作为备选方案，保证自己的实现适合所有用户，而随着时间的推移，在不改变当前实现的前提下，即可移除 JavaScript 和其他备选方案。

不过，在畅想未来之前，需要先对 HTML5 有个全面的了解。下一章中，很多新的结构标签在向你招手。还等什么，让我们去认识一下吧！

第 2 章

新的结构标签和属性

本书的前几章将着重讨论如何使用 HTML5 和 CSS 的特性来改善用户界面。从中，我们会了解到如何创建功能更强大的表单，如何轻松设计表格，以及如何提升页面在辅助设备上的可访问性。我们还会了解到如何使用内容生成[1]来改进打印样式表（print style sheet）的可用性。最后，我们将探索如何基于新的 contenteditable 属性进行在位编辑（in-place editing）。不过，在探索开始之前，让我们先看一下如何利用 HTML5 的新元素优化页面结构。

如今有一个严重的问题在影响着众多 Web 开发人员，即滥用 div。div 的滥用如同一种慢性病，Web 开发人员会使用多余的带有 ID 属性的 div 标签来作为元素的容器，这些 ID 值可能是 banner、sidebar、article、footer 等。div 滥用行为极具传染性。它在 Web 开发人员中的传染速度非常快，由于 div 不会在页面上直接显示，因此，轻度滥用的 div 可能会不为人知地隐藏数年之久。

下面是典型的 div 滥用：

```
<div id="navbar_wrapper">
  <div id="navbar">
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/">Home</a></li>
    </ul>
  </div>
</div>
```

上面的代码定义了一个无序列表，它是一个块元素[2]，包装它的两个 div 标签也是块元素。虽然包装元素上的 id 属性指明了其作用，但是去掉其中一个或多个包装元素后，我们仍能得到相同的效果。过度使用标记会导致代码臃肿，也不利于页面的美化和维护。

还好，这种现象有望得以改观。HTML5 规范提供了一组全新的语义化标签，它们可用于描

---

① 这里的“内容生成”是通过 CSS 的 content 属性来实现的。—— 译者注

② 谨记，块元素独占一行，而内联元素可以与其他元素在一行上。

述自身所包含的内容。许多开发人员都会在自己的设计中使用侧边栏、头部、尾部和区段（section），正因为如此，HTML5 规范引入了新的专用于划分页面不同逻辑区域的标签。我们可以在工作中使用这些新元素。有了 HTML5 的帮助，我们就能根除滥用 div 的现象了。

除了新的结构标签，我们还要谈谈 meter 元素，探讨如何使用新的自定义数据属性在元素中嵌入数据，进而取代惯用的类劫持（hijacking class）或现有属性。简言之，我们要弄清如何在适当的情况下选择合适的标签。

本章中，我们会探讨以下新元素及其特性。[①]

- <header>定义页面或区段的头部。[C5、F3.6、IE8、S4、O10]
- <footer>定义页面或区段的尾部。[C5、F3.6、IE8、S4、O10]
- <nav>定义页面或区段的导航区域。[C5、F3.6、IE8、S4、O10]
- <section>定义页面的逻辑区域或内容组合。[C5、F3.6、IE8、S4、O10]
- <article>定义正文或一篇完整的内容。[C5、F3.6、IE8、S4、O10]
- <aside>定义补充或相关内容。[C5、F3.6、IE8、S4、O10]
- 自定义数据属性（custom data attribute）——允许通过使用“data-”前缀向任意元素中添加自定义的数据属性。[所有浏览器都支持通过 JavaScript 的 getAttribute()方法获取这些属性。]
- <meter>描述指定范围内的数值。[C5、F3.5、S4、O10]
- <progress>用于显示实时进度的控件。[本书出版时，尚无浏览器支持。]

2.1 实例 1：用语义化标记重定义博客

在博客应用中，有大量内容需要使用结构化标记来进行组织。博客包括了头部、尾部、多种导航（博文归档链接、至其他博客或网站的链接列表和内部链接）、博文和回帖。下面，我们就以顶级优秀的 AwesomeCo 公司的博客首页为例，使用 HTML5 标记编写相应原型。

我们的设计方案如图 2-1 所示。这是一个非常典型的博客结构，其头部区域包含了水平导航。在主要的正文区域中，每篇文章都包括头部和尾部。此外，文章还可能包括摘要或旁白。侧边栏包含了其他的导航元素。最后是页面的尾部，它包含了联系方式和版权信息。结构本身并没有任何新意，唯一不同的是，我们将使用语义化标签来代替以往大量使用的 div 标签。

---

① 在后续的描述中，浏览器支持情况使用方括号加代码缩写和最低支持版本号的形式表示。缩写代码所表示的意义分别是：C 表示 Google Chrome，F 表示 Firefox，IE 表示 Internet Explorer，O 表示 Opera，S 表示 Safari，IOS 表示使用移动版 Safari 的 iOS 设备，A 表示 Android Browser。

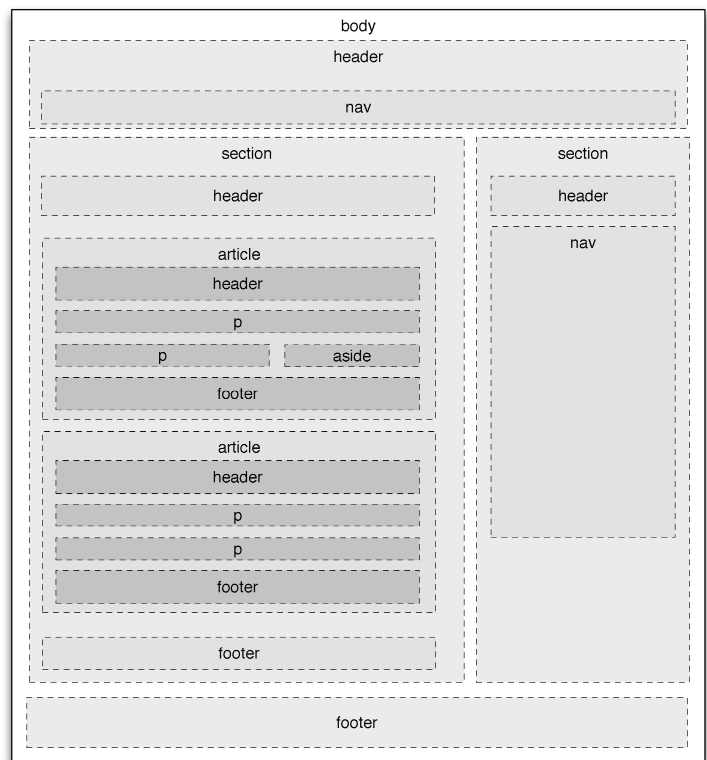


图 2-1 使用了 HTML5 语义化标记的博客结构

代码编写完成后的效果如图 2-2 所示。

AwesomeCo Blog!

Latest Posts Archives Contributors Contact Us

How Many Should We Put You Down For?

Posted by Brian on October 1st, 2010 at 2:39PM

The first big rule in sales is that if the person leaves empty-handed, they're likely not going to come back. That's why you have to be somewhat aggressive when you're working with a customer, but you have to make sure you don't overdo it and scare them away.

"Never give someone a chance to say no when selling your product."

One way you can keep a conversation going is to avoid asking questions that have yes or no answers. For example, if you're selling a service plan, don't ever ask "Are you interested in our 3 or 5 year service plan?" Instead, ask "Are you interested in the 3 year service plan or the 5 year plan, which is a better value?" At first glance, they appear to be asking the same thing, and while a customer can still opt out, it's harder for them to opt out of the second question because they have to say more than just "no."

25 Comments ...

Archives

- October 2010
- September 2010
- August 2010
- July 2010
- June 2010
- May 2010
- April 2010
- March 2010
- February 2010
- January 2010

© 2010 AwesomeCo.

Home About Terms of Service Privacy

图 2-2 完成之后的页面布局

2.1.1 以正确的文档类型声明为基础

若想使用 HTML5 的新元素，就需要让浏览器和验证器能够识别这些新标签。创建名为 index.html 的新页面，写入基本的 HTML5 模板：

Download html5newtags/index.html

```
Line 1  <!DOCTYPE html>
   2    <html lang="en-US">
   3      <head>
   4        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   5        <title>AwesomeCo Blog</title>
   6      </head>
   7
   8      <body>
   9      </body>
  10    </html>
```

注意示例中第 1 行的文档类型声明，它是 HTML5 的文档类型声明。如果经常编写网页，下面这段冗长难记的 XHTML 文档类型声明一定不会让你感到陌生：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

再来看看 HTML5 的文档类型声明：

```
<!DOCTYPE html>
```

相比之下，HTML5 的文档类型声明显然更加简单好记。

文档类型声明有两个作用。第一，验证器依据它来判断应该采用何种验证规则去验证代码。第二，文档类型声明能够强制 IE6、IE7 和 IE8 以“标准模式”（standards mode）渲染页面，在页面需要兼容所有浏览器时，这点极其重要。HTML5 文档类型声明具备了上述两种作用，它甚至可以被 IE6 识别。

2.1.2 头部

不要将头部（header）与 h1、h2、h3 这样的标题（heading）混为一谈，头部可能包含从公司的 Logo 到搜索框在内的各式各样的内容。目前，示例博客的头部只包含了博客的标题：

Download html5newtags/index.html

```
Line 1  <header id="page_header">
   2      <h1>AwesomeCo Blog!</h1>
   3    </header>
```

同一个页面中可以包含多个 header 元素。每个独立的区段或文章块都可以拥有自己的头部，

所以，示例代码中为头部添加可唯一标识一个元素的 ID 属性是有用的。借助唯一的 ID 值，开发人员可以更便捷地添加 CSS 样式，或使用 JavaScript 快速定位元素。

语义化标记

语义化标记的作用是描述内容。如果有多年的网页开发经验，那么你通常会将页面划分为若干区域，例如头部、尾部、侧边栏等，以便于在应用样式表和其他格式时，可以轻松识别出不同的页面区域。

语义化标记能够降低机器和开发人员理解内容含义和语境的难度。section、header、nav 等 HTML5 新标记的作用恰在于此。

2.1.3 尾部

footer 元素用来为文档或相邻区段定义尾部信息。你以前在网站上见过的页面尾部通常会包括版权日期和网站归属信息。HTML5 规范中允许在同一份页面文档中出现多个 footer 元素，这就意味着即使在博文中也能使用 footer 元素。

下面，我们来为页面定义一个简单的尾部。由于可以使用多个尾部，因此我们为页面尾部设置了 ID 属性。如同前面为头部设置 ID 属性一样，当需要为此元素及其子元素添加样式时，这种做法有助于我们准确识别它。

Download html5newtags/index.html

```
<footer id="page_footer">
  <p>&copy; 2010 AwesomeCo.</p>
</footer>
```

代码中的尾部只包含了版权日期。其实，尾部与头部类似，通常也会包含其他元素，如导航元素。

2.1.4 导航

导航对于网站的成功与否至关重要。如果访客难以找到所需信息，他们就不会再在网站停留，就这点而言，存在一个与导航相对应的 HTML 标签是有意义的。

我们要在文档头部添加导航。导航中链接分别指向博客的首页、文章列表页、博文作者列表页以及联系人信息页。

Download html5newtags/index.html

```
Line 1  <header id="page_header">
-         <h1>AwesomeCo Blog!</h1>
-         <nav>
-           <ul>
5             <li><a href="/">Latest Posts</a></li>
-             <li><a href="archives">Archives</a></li>
-             <li><a href="contributors">Contributors</a></li>
-             <li><a href="contact">Contact Us</a></li>
-           </ul>
10        </nav>
-       </header>
```

与 header 元素和 footer 元素一样，页面可以包含多个 nav 元素。通常情况下，头部和尾部都会包含导航，因此访客能够清晰地将其辨认出来。博客尾部的导航链接需要分别指向 AwesomeCo 的公司主页、“关于我们”（about us）页面以及“服务条款和隐私政策”页面。我们使用无序列表来组织这些链接，并将其置于页面的 footer 元素中：

Download html5newtags/index.html

```
<footer id="page_footer">
  <p>&copy; 2010 AwesomeCo.</p>
  <nav>
    <ul>
      <li><a href="http://awesomeco.com/">Home</a></li>
      <li><a href="about">About</a></li>
      <li><a href="terms.html">Terms of Service</a></li>
      <li><a href="privacy.html">Privacy</a></li>
    </ul>
  </nav>
</footer>
```

在随后的章节中，我们会用 CSS 来美化这两个导航条的外观，所以现在不必对此过于担心。HTML5 新元素的作用是描述内容，而非描述内容的外观。

2.1.5 区段和文章

区段是页面上的逻辑区域，在描述页面逻辑区域时，我们可以使用 section 元素来代替之前被随意滥用的 div 标签：

Download html5newtags/index.html

```
<section id="posts">
</section>
```

但是，也不要乱用 section 元素，要利用 section 元素将内容合理归类！上面的代码中，我们创建了用于容纳所有博文的区段。不过，单篇博文不适合都独立占用区段。为此，我们选择了更恰当的标签。

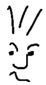
2.1.6　文章

最适合描述网页实际内容的元素非 article 标签莫属。页面上有很多元素，包括头部、尾部、导航、广告、部件（widget）、至其他博客或网站的链接（blogroll）、社交媒体书签等，这些繁杂的元素很容易让开发人员遗忘非常重要的一点——人们访问网站是源于对网站内容的兴趣，而 article 标签正好可以用来描述内容。

每篇文章都包括一个头部、一些内容和一个尾部，我们可以按下述方式定义一篇完整的文章：

Download html5newtags/index.html

```
<article class="post">
  <header>
    <h2>How Many Should We Put You Down For?</h2>
    <p>Posted by Brian on
      <time datetime="2010-10-01T14:39">October 1st, 2010 at 2:39PM</time>
    </p>
  </header>
  <p>
    The first big rule in sales is that if the person leaves empty-handed,
    they're likely not going to come back. That's why you have to be
    somewhat aggressive when you're working with a customer, but you have
    to make sure you don't overdo it and scare them away.
  </p>
  <p>
   One way you can keep a conversation going is to avoid asking questions
   that have yes or no answers. For example, if you're selling a service
   plan, don't ever ask &quot;Are you interested in our 3 or 5 year
   service plan?&quot; Instead, ask &quot;Are you interested in the 3
   year service plan or the 5 year plan, which is a better value?&quot;
   At first glance, they appear to be asking the same thing, and while
   a customer can still opt out, it's harder for them to opt out of
   the second question because they have to say more than just
   &quot;no.&quot;
  </p>
  <footer>
    <p><a href="comments"><i>25 Comments</i></a> ...</p>
  </footer>
</article>
```



小乔爱问……

section 标签和 article 标签有什么区别?

我们可以将 section 标签视为对文档逻辑部分的描述，而将 article 标签视为对具体内容的描述，例如杂志文章、博客日志、新闻条目等。

> 这些新标签描述了各自包含的内容。区段可以含有多篇文章，文章内部又可以划分若干区段。举例来说，区段就像报纸上的体育版，体育版中有多篇文章，每篇文章又可能分为若干部分。某些区段已经有了对应的标签，如头部和尾部。section 元素是更通用的元素，可以用来从逻辑上对其他元素进行分组。
>
> 语义化标记的作用正是传达内容的含义。

我们可以在 article 元素内部使用 header 元素和 footer 元素，以更方便地描述具体的逻辑区域，也可以用 section 元素将文档分为多个部分。

2.1.7　旁白和侧边栏

有时候，我们需要为主要内容添加一些附加信息，如引言、图表、补充观点、相关链接等。此时，可以使用新的 aside 标签来标识这些元素：

Download html5newtags/index.html

```
<aside>
  <p>
    &quot;Never give someone a chance to say no when
    selling your product.&quot;
  </p>
</aside>
```

我们将标注（callout quote）置于 aside 元素中，并将 aside 元素置于 article 元素内部，以保证 aside 元素能够紧邻着与其相关的内容。

带有 aside 元素的完整区段代码如下所示：

Download html5newtags/index.html

```
<section id="posts">
  <article class="post">
    <header>
      <h2>How Many Should We Put You Down For?</h2>
      <p>Posted by Brian on
        <time datetime="2010-10-01T14:39">October 1st, 2010 at 2:39PM</time>
      </p>
    </header>

    <aside>
      <p>
        &quot;Never give someone a chance to say no when
        selling your product.&quot;
      </p>
```

```
      </aside>
      <p>
        The first big rule in sales is that if the person leaves empty-handed,
        they're likely not going to come back. That's why you have to be
        somewhat aggressive when you're working with a customer, but you have
        to make sure you don't overdo it and scare them away.
      </p>
      <p>
       One way you can keep a conversation going is to avoid asking questions
       that have yes or no answers. For example, if you're selling a service
       plan, don't ever ask &quot;Are you interested in our 3 or 5 year
       service plan?&quot; Instead, ask &quot;Are you interested in the 3
       year service plan or the 5 year plan, which is a better value?&quot;
       At first glance, they appear to be asking the same thing, and while
       a customer can still opt out, it's harder for them to opt out of
       the second question because they have to say more than just
       &quot;no.&quot;
      </p>
      <footer>
        <p><a href="comments"><i>25 Comments</i></a> ...</p>
      </footer>
    </article>
  </section>
```

接下来，需要添加侧边栏了。

2.1.8　旁白绝非页面侧边栏

博客右侧是一个侧边栏，内含指向博文列表页的链接。倘若你认为可以使用 aside 标签来定义博客侧边栏就大错特错了。尽管可以那么做，但会违背规范的精神。aside 标签的设计初衷是为了展示与文章相关的内容。因此，它是呈现相关链接、术语表（glossary）或引言的最佳位置。

要将含有历史归档链接列表的侧边栏标记出来，使用 section 标签和 nav 标签即可。

Download html5newtags/index.html

```
<section id="sidebar">
  <nav>
    <h3>Archives</h3>
    <ul>
      <li><a href="2010/10">October 2010</a></li>
      <li><a href="2010/09">September 2010</a></li>
      <li><a href="2010/08">August 2010</a></li>
      <li><a href="2010/07">July 2010</a></li>
      <li><a href="2010/06">June 2010</a></li>
      <li><a href="2010/05">May 2010</a></li>
      <li><a href="2010/04">April 2010</a></li>
      <li><a href="2010/03">March 2010</a></li>
      <li><a href="2010/02">February 2010</a></li>
```

```
      <li><a href="2010/01">January 2010</a></li>
    </ul>
  </nav>
  </section>
```

以上是博客的 HTML 结构。接下来我们为新元素添加样式。

2.1.9　添加样式

我们可以像为 div 标签添加样式那样，来为新元素添加样式。首先，创建名为 style.css 的样式表文件，然后，在头部放置样式表引用以实现样式表文件与 HTML 文档的关联，代码如下：

Download html5newtags/index.html

```
<link rel="stylesheet" href="style.css" type="text/css">
```

首先，将页面内容整体居中，同时设置基本的字体样式：

Download html5newtags/style.css

```
body{
  width:960px;
  margin:15px auto;
  font-family: Arial, "MS Trebuchet", sans-serif;
}

p{
  margin:0 0 20px 0;
}
p, li{
  line-height:20px;
}
```

接下来，定义头部的宽度：

Download html5newtags/style.css

```
header#page_header{
  width:100%;
}
```

对导航链接应用样式，将无序列表变换成水平导航条：

Download html5newtags/style.css

```
header#page_header nav ul, #page_footer nav ul{
  list-style: none;
  margin: 0;
  padding: 0;
}
#page_header nav ul li, footer#page_footer nav ul li{
```

```
    padding:0;
    margin: 0 20px 0 0;
    display:inline;
  }
```

id 值为 posts 的区段需要浮动到页面左侧并保留一定宽度，文章内部的标注也需要浮动。此外，我们还要加大标注的字号：

Download html5newtags/style.css

```
section#posts{
  float: left;
  width: 74%;
}

section#posts aside{
  float: right;
  width: 35%;
  margin-left: 5%;
  font-size: 20px;
  line-height: 40px;
}
```

还需浮动侧边栏，并定义其宽度：

Download html5newtags/style.css

```
section#sidebar{
  float: left;
  width: 25%;
}
```

可度量的数值范围表和进度条

如果需要在 Web 应用中展示抵押品价值范围表或实现上传进度条，可以考虑使用 HTML5 规范中引入的 meter 元素和 progress 元素。

对于已知最小值和最大值的数值范围，我们可以借助 meter 元素对其上的固定一点进行语义描述。meter 元素的使用应符合 HTML5 规范的要求，所以不应用 meter 元素来描述如高度、重量等没有最小值或最大值的特征，除非已为它们设定了明确的数值范围。例如，有一个募捐网站，我们希望显示募捐款距离 5000 美元的目标还有多远，代码如下：

Download html5_meter/index.html

```
<section id="pledge">
  <header>
    <h3>Our Fundraising Goal</h3>
  </header>
  <meter title="USD" id="pledge_goal"
         value="2500" min="0" max="5000">
```

```
      $2500.00
    </meter>
    <p>Help us reach our goal of $5000!</p>
  </section>
```

progress 元素与 meter 元素极为相似，其设计初衷是显示实时进度，比如显示上传文件时的进度。相比较之下，meter 元素的设计初衷是显示当前相对固定的可度量值，如某用户在服务器上的可用存储空间的快照。progress 元素的用法与 meter 元素的用法类似：

Download html5_meter/progress.html

```
<progress id="progressbar" max=100><span>0</span>%</progress>
```

尽管还没有浏览器为 meter 元素和 progress 元素定义默认样式，但是我们能够使用 JavaScript 获取 meter 的数值并创建自己的可视效果，可以用 meter 元素或 progress 元素对数据进行语义化描述。参考本书中与 meter 元素相关的示例文件，你就能够掌握其用法。

我们需要定义尾部样式，并在尾部上清除浮动，从而保证尾部始终位于页面底部，代码如下：

Download html5newtags/style.css

```
footer#page_footer{
  clear: both;
  width: 100%;
  display: block;
  text-align: center;
}
```

以上是基本的样式。至此，我相信你会将它的效果做得越来越好。

2.1.10　回退

虽然博客首页在 Firefox、Chrome 和 Safari 中显示正常，但当管理人员通过 IE 浏览到一个乱七八糟的页面时，他们难免会失望。内容倒是呈现出来了，但由于 IE 不兼容 HTML5 的新元素，所以无法对它们应用样式，整个页面看起来就像 20 世纪 90 年代中期的作品。

唯一能让 IE 在新元素上应用样式的方法是使用 JavaScript 将新元素定义为文档的一部分。复杂的事情瞬间变容易了。我们只需在页面的 head 标签内添加 hack 代码即可，这是为了保证代码的执行能够先于浏览器对其他元素的渲染。我们将 hack 代码置于只有 IE 浏览器才能识别的条件注释（conditional comment）中：

Download html5newtags/index.html

```
<!--[if lt IE 9]>
<script type="text/javascript">
  document.createElement("nav");
```

```
    document.createElement("header");
    document.createElement("footer");
    document.createElement("section");
    document.createElement("aside");
    document.createElement("article");
  </script>
  <![endif]-->
```

上面这种特殊的注释是针对 IE 9.0 以前的所有版本的。现在重新加载页面，一切正常。

我们创建的页面依赖于 JavaScript，这点需要开发人员考虑清楚。文档的组织形式和可读性得到了一定程度的改观，由此来看，这么做是值得的。同时，内容仍可以正常显示且能被屏幕阅读器阅读，因而没有“可访问性”的顾虑。不过，对于那些故意禁用 JavaScript 的用户来说，你的做法貌似极其“过时”。

本节的方法展示了如何为少量元素添加支持以及具体的做法。Remy Sharp 杰出的 HTML-Shiv[①]对此进行了更为深远的研究，如果你希望支持更多元素，HTML-Shiv 方法更合适。

2.2 实例 2：使用自定义数据属性创建弹出窗口

在开发 Web 应用时，有时会用 JavaScript 获取文档之外的信息，某些情况下，我们需要用一些技巧来处理这些额外信息以保证 Web 应用能够正常运行。一般而言，技巧无外乎是将额外的信息塞入事件处理程序或滥用 `rel` 属性或 `class` 属性以方便注入行为。感谢 HTML5 规范引入了自定义数据属性，让不堪回首的往日一去不返。

所有的自定义数据属性都以 `data-`前缀开头，HTML5 文档的验证器会在验证时忽略它。开发人员可以在任意元素中加入自定义数据属性，属性值可以是照片的元数据、经纬度坐标或者弹出窗口的尺寸。最棒的是，几乎在所有浏览器中，你都能够使用自定义数据属性，因为我们可以轻易地使用 JavaScript 来获取它们。

2.2.1 行为与内容的分离，或者说为什么设置onclick不好

过去的数年里，弹出窗口的名声一直不好，而这通常是很自然的事情。弹出窗口常被用来向用户显示广告，或者伪装成不被怀疑的间谍软件或病毒，而最糟糕的是骗取个人信息并随后加以出售。因此，大多数浏览器都有阻止弹出窗口的插件也就不足为怪了。

不过，弹出窗口并非总是不好。Web 应用的开发人员常常需要依赖于弹出窗口，以便向用户显示在线帮助信息、附加选项或者其他重要的用户界面功能。为了不让弹出窗口过于惹人厌烦，

① 参见 http://code.google.com/p/html5shiv/。

我们需要用一种不引人注目的方式来实现它。浏览 AwesomeCo 公司的人力资源页面时，你会看到几个链接，它们用于打开显示服务条款的弹出窗口，其代码实现大多如下：

Download html5_popups_with_custom_data/original_example_1.html

```
<a href='#'
  onclick="window.open('holiday_pay.html',WinName,'width=300,height=300);">
  Holiday pay
</a>
```

通过链接来触发弹出窗口的方式颇为常见。事实上，JavaScript 菜鸟在初学如何实现弹出窗口时大都会采用这种方式。在进一步实现所需效果之前，我们要指出此种方式的两个问题。

2.2.2　提升可访问性

链接的目标地址没有设置！如果 JavaScript 被禁用了，那么链接将无法引导用户进入相应页面。这是一个非常严重的问题，以至于我们需要立即解决。我建议开发人员永远不要省略 `href` 属性，任何情况下都不要为 `href` 属性赋“#”及类似值。现在，我们将弹出窗口中显示的资源的地址赋值给 `href` 属性，代码如下：

Download html5_popups_with_custom_data/original_example_2.html

```
<a href='holiday_pay.html'

  onclick="window.open(this.href,WinName,'width=300,height=300);">
  Holiday pay
</a>
```

上面的代码中，我们通过 JavaScript 代码读取 `a` 元素的 `href` 属性值，进而得到资源的链接地址。

构建可访问的页面，第一步是确保禁用 JavaScript 的情况下，所有功能仍能正常运转。

2.2.3　废弃onclick

注意保持行为与内容分离，正如用链接样式表保持样式信息分离一样。开始的时候，使用 `onclick` 会带来便利，但是想象一下页面上有 50 个链接的情况，那时你会看到 `onclick` 方法失控的场面。你将只能一遍又一遍地编写重复的 JavaScript 代码。如果是通过服务器端代码来生成浏览器端代码，那么你就是在增加 JavaScript 事件，进而会导致大量不必要的 HTML 代码的出现。

替代方法是为每个链接分配可识别的 CSS 类名：

Download html5_popups_with_custom_data/original_example_3.html

```
<a href="holiday_pay" class="popup">Holiday Pay</a>
```

Download html5_popups_with_custom_data/original_example_3.html

```
var links = $("a.popup");

links.click(function(event){
    event.preventDefault();
    window.open($(this).attr('href'));
});
```

上述代码中使用了 jQuery 的选择器来获取类名为 popup 的元素，随后，我们为其中每个元素的 click 事件分别添加一个监听器。当有人单击链接时，click 方法中的代码会被执行。preventDefault 方法用于阻止默认的单击事件行为。在示例中，它阻止了页面跳转。

我们还忘了一件事——没有设置窗口的尺寸信息和位置信息，而在未优化的代码中，我们是设置了的。我们希望即使是不太熟悉 JavaScript 代码的页面设计者，也能基于每个链接设置窗口尺寸。

2.2.4 自定义数据属性来解围

当创建应用了 JavaScript 的 Web 应用时，刚刚提到的情况[①]比较常见。如你所见，在代码中存储窗口的期望高度和宽度是可取的，但是 onclick 方法有诸多弊端。我们可以改换在元素上嵌入属性的方式加以实现。现在要做的是将链接改造成下面这种形式：

Download html5_popups_with_custom_data/popup.html

```
<a href="help/holiday_pay.html"
   data-width="600"
   data-height="400"
   title="Holiday Pay"
   class="popup">Holiday pay</a>
```

离完成只有一步之遥，修改之前编写的 click 事件以抓取链接上设置的各项自定义数据属性，然后将其传入 window.open 方法。

Download html5_popups_with_custom_data/popup.html

```
$(function(){
  $(".popup").click(function(event){
    event.preventDefault();
    var href = $(this).attr("href");
```

① 即不熟悉 JavaScript 的页面设计者能够借助自己熟悉的技术去影响应用的展示效果。——译者注

```
    var width = $(this).attr("data-width");
    var height = $(this).attr("data-height");
    var popup = window.open (href,"popup",
      "height=" + height +",width=" + width + "");
  });
});
```

收工！现在，单击链接后会打开一个新窗口。

> **提醒**
>
> 在示例中，我们使用自定义数据属性为客户端脚本提供了额外的信息。对于具体问题而言，这是一种聪明的做法，同时也说明了自定义数据属性的使用方法。尽管这种做法倾向于将表现信息与标签相混合，但是它向你展示了使用 JavaScript 读取嵌入在页面中的数据是多么的简单。

2.2.5 回退

只要浏览器支持 JavaScript，自定义数据属性就能正常工作。自定义数据属性不会使浏览器出错，同时，HTML5 文档类型声明可保证文档是有效的，因为以 `data-`开头的属性都会被忽略。

2.2.6 未来展望

一旦新标签和属性得到了广泛支持，我们就可以用它们来做一些有趣的事情。使用打印样式表，我们能够轻易地识别并隐藏导航和文章的尾部：

```
nav, article>footer{display:none}
```

借助于脚本语言，我们能够迅速识别网站内或页面中的所有文章。但最重要的是，我们用到的标签能够恰如其分地描述其所标记的内容，这样一来，我们就能够写出更好的样式表和更强大的 JavaScript 代码了。

开发人员可以用自定义数据属性将各式各样的信息嵌入标记中。事实上，在第 6 章中，我们会再次用到自定义数据属性。

借助于自定义数据属性，使用 JaveScript 定位到所有 `data-remote=true` 的表单标签，就能得知哪些表单标签应使用 Ajax 进行提交，这种做法与 Ruby on Rails 框架中的做法是一致的。

我们还可以使用自定义数据属性将日期和时间缓存在页面中，并以用户时区为基准来显示日期和时间。只需将时间以 UTC 的形式置于 HTML 页面中，在客户端将其转换成用户本地时间即可。自定义数据属性允许开发人员在页面中嵌入真实有用的数据，而越来越多的框架和库会利用它的优点。我确信，在日常工作中，你会发现它有很多用处。

至此，我们可以一劳永逸地根除滥用 div 的情况了。

第 3 章

创建易用的 Web 表单

如果曾设计过复杂的用户界面，你就会知晓基本 HTML 表单控件的局限性有多大。为了实现复杂的表单，你不得不使用文本框、选择菜单、单选按钮、复选框。某些情况下甚至还需要动用笨重的多选列表，为此你需要不厌其烦地反复向用户解释如何使用它。（“按住 Ctrl 键，单击所需的列表项，如果是 Mac，则按住 Cmd 键再点选。”）

所以，你会像所有优秀 Web 开发人员所做的那样，转而使用 Prototype 或 jQuery 库，或者维护一套整合了 HTML、CSS 和 JavaScript 的控件和功能。但是，当表单上同时出现了滑块、日历控件、选值框（spinbox）、具有自动完成功能的文本框和可视化编辑器时，你就会迅速意识到你为自己制造了一场梦魇。此时，你不仅需要保证自己在当前页面上引用的控件不会与已引用的其他控件发生冲突，而且还要保证它们不会与其他 JavaScript 库相冲突。你可能会花费数小时来调试日期选择器，并最终发现是 Prototype 库出了问题，而根本原因则是因为 jQuery 库覆盖了 Prototype 库的`$()`函数。于是，改用 jQuery 的 `noConfilct()`方法，完成改动后，你又发现颜色选择器无法正常工作了，因为插件代码编写得不够健壮。

看到这里，如果你笑了，说明你遇到过上述情况。如果你生气了，我猜也是因为同样的原因。还好，有希望改变这一切。本章中，我们将使用一些新的表单域类型来构建几个 Web 表单，并同时实现自动聚焦功能和占位文本。

最后，我们将讨论如何使用新的 `contenteditable` 属性将任意 HTML 域转换成用户输入控件。

具体而言，本章会覆盖以下功能。

- 电子邮件输入域[`<input type="email">`]——显示一个用于输入电子邮件地址的表单域。[O10.1、IOS]
- URL 输入域[`<input type="url">`]——显示一个用于输入 URL 的表单域。[O10.1、IOS]
- 电话号码输入域[`<input type="tel">`]——显示一个用于输入电话号码的表单域。[O10.1、IOS]

- 搜索域[<input type="search">——显示一个用于输入搜索关键字的表单域。[C5、S4、O10.1、IOS]
- 滑块（范围选择）[<input type="range">]——显示一个滑块控件。[C5、S4、O10.1]
- 数值设定框 [<input type="number">]——显示一个用于输入数值的表单域，通常是选值框。[C5、S5、O10.1、IOS]
- 日期选择域[<input type="date">]——显示用于日期选择的表单域，支持日期、月份或周。[C5、S5、O10.1]
- 可以选择时间的日期选择域[<input type="datetime">]——显示一个用于选择日期和时间的表单域，支持日期和时间、本地日期和时间或只有时间。[C5、S5、O10.1]
- 颜色选择域[<input type="color">]——显示一个用于指定颜色的表单域。[C5、S5]（Chrome 5 和 Safari 5 支持此类型，但并不将其显示为特殊的控件。）
- 支持自动聚焦[<input type="text" autofocus>]——支持将焦点置于特定的表单元素上。[C5、S4]
- 支持占位文本[<input type="email" placeholder="me@example.com">]——支持在表单域内显示占位文本。[C5、S4、F4]
- 支持在位编辑 [<p contenteditable>lorem ipsum</p>]——支持通过浏览器在位编辑内容。[C4、S3.2、IE6、O10.1]

现在，我们来学习其中一些非常有用的表单域类型。

3.1　实例 3：使用新的输入域描述数据

HTML5 引入了几种新的输入类型，开发人员可以用它们来更加精确地描述用户输入数据的类型。除了标准的文本框、单选按钮和复选框，你还可以使用电子邮件输入框、日期选择器、颜色选择器、数值设定框和滑块。浏览器能够基于这些新的表单域类型为用户渲染出效果更好的控件，而在这一过程中不需要编写任何的 JavaScript 代码。移动设备、平板电脑上的虚拟键盘以及触摸屏能够根据表单域类型的不同展示出不同的键盘布局。例如，当用户在 URL 类型或 email 类型的文本框中输入数据时，iPhone 上移动版的 Safari 浏览器会显示另一种键盘布局，以方便用户输入诸如@、.、:、/这样的特殊字符。

3.1.1　改进AwesomeCo项目中的表单

AwesomeCo的目标是创建一个全新的项目管理类的Web应用，以方便开发人员和项目经理把控其所参与的众多项目的进度。每个项目都有自己的名称、电子邮件地址和测试环境的URL，通

过测试环境的URL，项目经理可以在项目开发过程中预览网站的效果。此外，项目属性还包括启动的日期、优先级以及估算的项目完成所需小时数。最后，负责开发的经理往往倾向于为每个项目分配一种颜色，以方便其在查看项目报告时快速识别出不同的项目。

下面，我们使用新的 HTML5 表单控件来创建项目首选项页面的示例。

3.1.2 创建基础表单

先创建用于发送 POST 请求的基础表单。因为项目名称输入框没有什么特殊要求，所以我们使用 `text` 类型的文本框。

Download html5forms/index.html

```
<form method="post" action="/projects/1">

  <fieldset id="personal_information">
    <legend>Project Information</legend>
    <ol>
      <li>
        <label for="name">Name</label>
        <input type="text" name="name" autofocus id="name">
      </li>
      <li>
        <input type="submit" value="Submit">

      </li>
    </ol>

  </fieldset>

</form>
```

注意，我们在有序列表中放置了用于标记表单的 `label` 标签。创建具有良好可访问性的表单时，描述性标签 `label` 必不可少。`label` 标签的 `for` 属性用于引用与其相关联的表单元素的 `id`。这种用法有助于屏幕阅读器识别页面上的表单域。有序列表提供了一种良好的展现形式来列出表单域，从而避开了复杂的表格结构或 `div` 结构。此外，有序列表能够辅助标记出你所期望用户填写表单域的顺序。

3.1.3 使用range类型创建滑块

滑块常被用户用于增减数值。一个实用场景是项目经理使用滑块来快速直观地修改项目优先级。在 HTML5 中，你可以使用 `range` 类型来实现滑块。

```
Download html5forms/index.html
<label for="priority">Priority</label>
<input type="range" min="0" max="10"
       name="priority" value="0" id="priority">
```

仿照 3.1.2 节的代码，用 `li` 元素把上面的代码包起来，然后将其加入到基础表单中。

Chrome 和 Opera 浏览器都支持滑块部件，其效果如图 3-1 所示。

Priority


图 3-1

注意，我们在代码中设定了 `min` 属性和 `max` 属性来限制滑块域值的范围。

3.1.4 使用选值框处理数字

我们在很多场景中都会用到数字，尽管直接输入数字的方式很简单，但使用选值框（spin box）可以使数字微调变得更容易。作为一种控件，选值框带有用于增减数值的箭头。下面用选值框来实现估算时间的功能。以这种方式实现后，调整时间更容易。

```
Download html5forms/index.html
<label for="estimated_hours">Estimated Hours</label>
<input type="number" name="estimated_hours"
       min="0" max="1000"
       id="estimated_hours">
```

Opera 支持选值框控件，其效果如图 3-2 所示。

Estimated Hours


图 3-2

默认情况下，也可以在选值框中直接输入数字，与 `range` 类型的滑块类似，开发人员可以为其设定最大值和最小值。但是，直接输入的数字不受最小值和最大值的限制。

此外，控件增减的步长是由 `step` 属性来指定的。`step` 属性可以是任意数字，其默认值为 1。

3.1.5 日期控件

记录项目开始时间非常重要，我们希望其实现越简单越好。此时，`date`类型的日期控件是最佳选择。

Download html5forms/index.html

```
<label for="start_date">Start date</label>
<input type="date" name="start_date" id="start_date"
       value="2010-12-01">
```

写作本书的时候，Opera 是唯一支持完整日期选择控件的浏览器。

实现后的效果如图 3-3 所示。

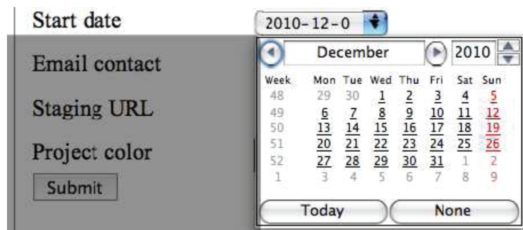


图　3-3

Safari 5.0 的显示效果类似于选值框，带有增减日期的箭头。如果值为空，日期控件会默认显示“1582”。其他浏览器会将日期控件渲染成一个文本框。

3.1.6　email类型

在 HTML5 规范中，email 类型文本框被设计成用于输入单一的电子邮件地址或者电子邮件地址列表，因此，在需要输入电子邮件的场合使用它是最合适不过的了。

Download html5forms/index.html

```
<label for="email">Email contact</label>
<input type="email" name="email" id="email">
```

移动设备还能从 email 类型的表单域中获得更多好处，只需改变虚拟键盘的布局，电子邮件地址的输入就会变得更容易。

3.1.7　url类型

这种表单域类型用于处理 URL。如果访客使用的是 iPhone，它会特别好用，因为 iPhone 会为其显示相应的键盘布局，上面有一些快捷按钮辅助用户快速输入网址，类似于在 Safari 移动版的地址栏中输入 URL 时显示的键盘。添加用于输入临时 URL 的文本框，代码如下：

Download html5forms/index.html

```
<label for="url">Staging URL</label>
<input type="url" name="url" id="url">
```

使用 url 类型时，虚拟键盘的布局也会发生变化。

3.1.8　color类型

最后，我们需要能够输入颜色码。为此，我们使用 color 类型。

Download html5forms/index.html

```
<label for="project_color">Project color</label>
<input type="color" name="project_color" id="project_color">
```

编写本书时，还没有浏览器能够显示颜色选择器控件，但这并不应成为阻止你使用 color 类型的理由。运用合适的标记来描述内容，会让你在未来游刃有余，特别是在需要提供兼容性支持的时候。

目前，Opera 能够支持上述新控件中的大部分，如图 3-4 所示。不过，使用 Firefox、Safari 或者 Google Chrome 浏览页面的时候，你只会看到些许不同。接下来，我们要修正浏览器的兼容性问题。

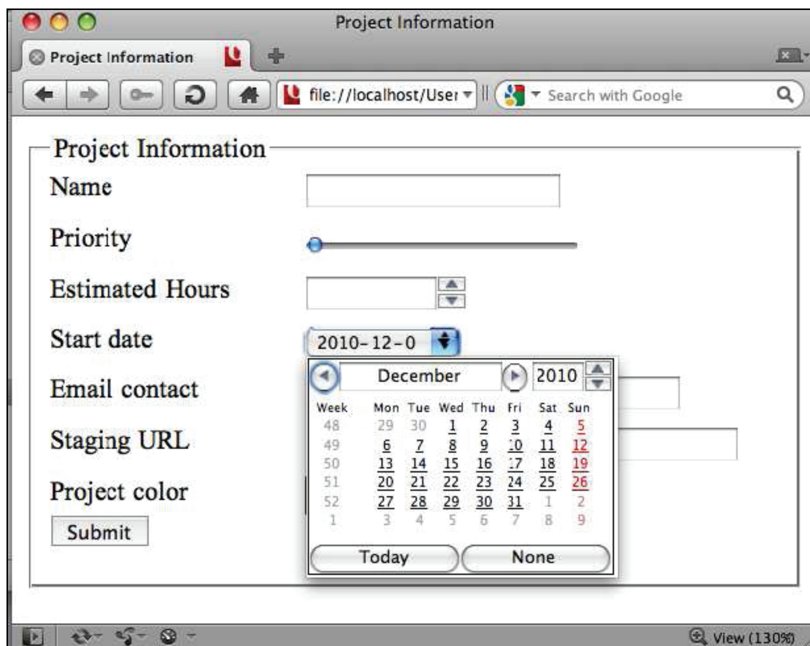


图 3-4　Opera 已经支持的一些表单控件

3.1.9　回退

对于无法解析的表单域类型，浏览器会将其视为 text 类型，所以刚刚制作的表单仍然可用。

在这一点上，开发人员可以绑定 jQuery UI 或 YUI 部件来对新控件进行变换。随着时间的推移，越来越多的浏览器会支持新控件，介时，你就可以移除这些 JavaScript 钩子了。

3.1.10 替换颜色选择器

借助 jQuery 的 CSS3 属性选择器，我们能够轻易地识别并替换颜色选择器。定位出所有 `color` 类型的 `input` 表单域，并对其应用一个名为 SimpleColor 的 jQuery 插件：

Download html5forms/index.html

```
if (!hasColorSupport()){
  $('input[type=color]').simpleColor();
}
```

由于在标记中使用了新的表单类型，所以我们无需添加额外用于标识颜色选择器的类名或其他标记。属性选择器与 HTML5 配合得天衣无缝。

如果浏览器原生支持颜色选择器，那就没必要调用颜色选择器插件了，为此要用一些 JavaScript 代码来检测浏览器是否支持 `color` 类型的 `input` 表单域。

Download html5forms/index.html

```
Line 1  function hasColorSupport(){
   -      input = document.createElement("input");
   -      input.setAttribute("type", "color");
   -      var hasColorType = (input.type !== "text");
   5      //处理 Safari/Chrome 的部分实现
   -      if(hasColorType){
   -        var testString = "foo";
   -        input.value=testString;
   -        hasColorType = (input.value != testString);
  10      }
   -      return(hasColorType);
   -    }
```

首先，使用 JavaScript 原生方法创建 `input` 元素并将其 `type` 属性值置为 `color`。接下来，取出 `type` 属性值以确定浏览器是否允许我们将其值置为 `color`。如果返回值是 `color`，则说明浏览器支持 `color` 类型，否则需要我们应用插件来完成相应功能。

代码的第 6 行，有趣的事情发生了。Safari 5 和 Google Chrome 5 部分实现了 `color` 类型，它们支持 `type` 属性值为 `color`，却不会实际显示颜色选择部件。最终，我们在页面上获得的还只是一个文本框。为此，我们在检测方法中为表单域的 `value` 属性赋值，看看值是否会被替换。如果被替换，我们就可以认为浏览器实现了对颜色选择器的支持，因为输入域的行为与普通文本框的行为已大不相同。

完整的替换颜色选择器的实现代码如下：

Download html5forms/index.html

```
if (!hasColorSupport()){
  $('input[type=color]').simpleColor();
}
```

解决方案虽然有效，但不够健壮。它仅针对一组特定的浏览器，而且仅解决了颜色控件的浏览器兼容性问题。其他控件也都有各自的解决方案等待你去逐个学习。感谢上帝，有个不错的解决方案可以毕其功于一役。

3.1.11　Modernizr

Modernizr[①]库能够检测浏览器是否支持 HTML5 以及 CSS3 的许多特性。尽管它没有为缺失的功能提供替代方案，但就检测表单域而言，它提供了几套与之前实现的解决方案相似，但更加可靠的机制。

在项目中引入 Modernizr 之前，你要确保已经了解其运行机制。无论是否亲自编码，只要你在项目中引入了 Modernizr，就应该对它负责。Modernizr 目前尚无法处理 Safari 对颜色选择器的部分支持。这样一来，当 Chrome 或 Firefox 的下一版本发布时，你可能不得不为它们提供一套替代方案。谁知道呢，也许到时候，你会将自己的解决方案贡献给 Modernizr！

可以用类似的方法实现日期选择器、滑块等控件的替代方案。jQuery UI 库[②]中以组件的形式包含了滑块和日期选择器。在页面上引用 jQuery UI 库，检测浏览器是否原生支持控件，如果不支持，就用 JavaScript 版控件进行替换。

最终，你会逐步停止使用 JavaScript 控件，完全依靠浏览器内置的控件。浏览器兼容性的检测颇为复杂，因此在开发过程中，Modernizer 会非常有用。不过，在本书余下的章节中，为了方便读者了解相应的运作机理，我们仍然会手动编写兼容性检测代码。

除了新的表单域类型外，HTML5 还引入了一些其他提升可用性的表单域属性。下面，我们来了解一下 autofocus 属性。

3.2　实例 4：使用 autofocus 属性定位第一个表单域元素

客户端完成加载页面时，将光标定位到表单的第一个表单域上能够提高用户输入数据的速

① 参见 http://www.modernizr.com/。

② 参见 http://jqueryui.com。

度。许多搜索引擎通过 JavaScript 实现了这个功能，而现在，HTML5 提供了自动聚焦功能，并将其视为语言的一部分。

要使用 HTML5 的自动聚焦功能，你只需在任意表单域中添加 autofocus 属性即可，如同我们在 3.1 节所做的那样。

Download html5forms/index.html

```
<label for="name">Name</label>
<input type="text" name="name" autofocus id="name">
```

autofocus 属性稳定生效的前提是页面中设置了唯一的 autofocus 属性。如果设置了多个，浏览器会把用户光标定位到最后一个设置了 autofocus 属性的表单域上。

回退

仅需少量 JavaScript 代码即可检测出是否存在 autofocus 属性，如果用户浏览器不支持 autofocus 属性，则可使用 jQuery 完成元素聚焦。这也许是你能提供的最简单的替代方案。

Download html5forms/autofocus.js

```
function hasAutofocus() {
  var element = document.createElement('input');
  return 'autofocus' in element;
}

$(function(){
   if(!hasAutofocus()){
     $('input[autofocus=true]').focus();
   }
});
```

在页面中植入上面这段 JavaScript 代码后，你就可以随时使用 autofocus 属性而不必担心浏览器兼容性问题了。

页面加载完成时，autofocus 属性能够略微简化用户开始填写表单的过程。现在，你可能还想在表单域上提供一些与类型相关的信息，如希望用户输入什么类型的值。没问题，接下来的 placeholder 属性正好可以解决这个问题。

3.3 实例 5：使用 placeholder 属性进行提示

占位文本向用户说明了他们应该如何填写页面上的表单域。图 3-5 是一张带有占位文本的注册表单。下面，我们来搭建它。

Create New Account
First Name
'John'
Last Name
'Smith'
Email
user@example.com
Password
8-10 characters
Password Confirmation
Type your password
Sign Up

图 3-5　占位文本能够帮助用户理解他们需要输入怎样的内容

3.3.1　简单的注册表单

AwesomeCo 的支持站点要求用户注册账号，其中最大的问题之一是用户会在注册过程中多次尝试设置安全性较低的密码。下面，让我们用占位文本向用户说明密码的安全性要求。考虑到一致性，我们为其他表单域也添加了占位文本。

在各个输入域上添加 `placeholder` 属性即可实现占位文本，代码如下：

Download **html5placeholdertext/index.html**

```
<input id="email" type="email"
       name="email" placeholder="user@example.com">
```

表单中的域均带有 `placeholder` 属性，整个表单的标记代码如下：

Download **html5placeholdertext/index.html**

```
<form id="create_account" action="/signup" method="post">
  <fieldset id="signup">
    <legend>Create New Account</legend>
    <ol>
```

```
      <li>
        <label for="first_name">First Name</label>
        <input id="first_name" type="text"
               autofocus="true"
               name="first_name" placeholder="'John'">
      </li>
      <li>
        <label for="last_name">Last Name</label>
        <input id="last_name" type="text"
               name="last_name" placeholder="'Smith'">
      </li>
      <li>
        <label for="email">Email</label>
        <input id="email" type="email"
               name="email" placeholder="user@example.com">
      </li>
      <li>
        <label for="password">Password</label>
        <input id="password" type="password" name="password" value=""
               autocomplete="off" placeholder="8-10 characters" />
      </li>
      <li>
        <label for="password_confirmation">Password Confirmation</label>
        <input id="password_confirmation" type="password"
               name="password_confirmation" value=""
               autocomplete="off" placeholder="Type your password again" />
      </li>
      <li><input type="submit" value="Sign Up"></li>
    </ol>
  </fieldset>
</form>
```

3.3.2 阻止自动完成

可能你已经注意到了，我们在表单的密码框上设置了`autocomplete`属性。HTML5引入了`autocomplete`属性，用以通知浏览器不要为当前表单域自动填充数据。某些浏览器能够记录用户之前输入的数据，而在某些场合下，我们想告知浏览器我们不希望用户使用记录数据。

由于再次用到了有序列表元素来组织表单域，因此，需要添加一点基础CSS样式以让表单看起来更漂亮。

Download **html5placeholdertext/style.css**

```
fieldset{
  width: 216px;
}

fieldset ol{
```

```
  list-style: none;
  padding:0;
  margin:2px;
}

fieldset ol li{
  margin:0 0 9px 0;
  padding:0;
}

/* 使输入独立成行 */
fieldset input{
  display:block;
}
```

现在，Safari、Opera 和 Chrome 的用户能够看到表单域中的帮助文本了。接下来，要让 Firefox 和 IE 用户也能看到。

3.3.3 回退

借助于 JavaScript 在表单域中放置占位文本并不需要太多工作量。用代码测试每个表单域的 `value` 属性，如果为空则将其设为占位文本。当焦点落在表单域上时，将其值置为空，而当表单域失去焦点时，再测试一遍 `value` 属性。如果值不同则什么都不做，否则，将其值再次置为占位文本。

检测浏览器是否支持 `placeholder` 属性的代码与之前检测 `autofocus` 属性支持度的代码类似：

Download **html5placeholdertext/index.html**

```
function hasPlaceholderSupport() {
  var i = document.createElement('input');
  return 'placeholder' in i;
}
```

接下来，编写 JavaScript 代码来处理相应逻辑。解决方案的基础是 Andrew January[①]以及其他一些人的工作。我们会用存储在 `placeholder` 属性中的文本来填充各个表单域的 `value` 属性值。当用户选择某个表单域时，我们将移除占位文本。让我们将上述逻辑封装成一个 jQuery 插件，以便应用于表单。至于如何让插件工作，可以参看本节随后部分中的 jQuery 插件说明。

---

① 原始脚本位于http://www.morethannothing.co.uk/wp-content/uploads/2010/01/placeholder.js，但是它不支持 IE 下的密码框。

Download html5placeholdertext/jquery.placeholder.js

```
(function($){

  $.fn.placeholder = function(){

    function valueIsPlaceholder(input){
      return ($(input).val() == $(input).attr("placeholder"));
    }
    return this.each(function() {

      $(this).find(":input").each(function(){

        if($(this).attr("type") == "password"){

          var new_field = $("<input type='text'>");
          new_field.attr("rel", $(this).attr("id"));
          new_field.attr("value", $(this).attr("placeholder"));
          $(this).parent().append(new_field);
          new_field.hide();

          function showPasswordPlaceHolder(input){
            if( $(input).val() == "" || valueIsPlaceholder(input) ){
              $(input).hide();
              $('input[rel=' + $(input).attr("id") + ']').show();
            };
          };

          new_field.focus(function(){
            $(this).hide();
            $('input#' + $(this).attr("rel")).show().focus();
          });
          $(this).blur(function(){
             showPasswordPlaceHolder(this, false);
          });

          showPasswordPlaceHolder(this);

        }else{

          // 用占位文本替换其值
          // 可选的 reloab 参数用来解决 Firefox 和
          // IE 缓存域值的问题
          function showPlaceholder(input, reload){
            if( $(input).val() == "" ||
              ( reload && valueIsPlaceholder(input) ) ){
                $(input).val($(input).attr("placeholder"));
              }
          };

```

```
50              $(this).focus(function(){
 -                if($(this).val() == $(this).attr("placeholder")){
 -                  $(this).val("");
 -                };
 -              });
55
 -              $(this).blur(function(){
 -                 showPlaceholder($(this), false)
 -              });
 -
60
 -              showPlaceholder(this, true);
 -            };
 -          });
 -
65          // 禁止表单提交默认值
 -          $(this).submit(function(){
 -            $(this).find(":input").each(function(){
 -              if($(this).val() == $(this).attr("placeholder")){
 -                $(this).val("");
70              }
 -            });
 -          });
 -
 -        });
75      };
 -
 -    })(jQuery);
```

这一插件有几个有趣的地方，你应该了解一下。代码第 45 行，如果表单域值为空或者用户刷新了页面，我们会为表单域重新加载占位文本。Firefox 和其他一些浏览器会持久化表单中的数据，此时，表单域的值已被置成了占位文本，我们可不希望用户意外地将占位文本提交给服务器。因此，加载页面时，我们给这一方法传递了一个 `true` 值，见代码第 61 行。

密码框的行为与其他表单域有些许差异，我们需要对此进行不同处理。看一下代码第 12 行，我们在检测文本域是否为密码框，如果是则将 `type` 类型值置为常规的 `text`，即普通的文本框，这样一来，显示的值将不再是星号。修改密码框的 `type` 属性时，某些浏览器会抛出异常，因此我们采用了更安全的做法——交替显示密码框和用于显示占位文本的文本域。根据用户与表单域的交互行为来决定哪个表单域应该显示，而哪个表单域应该隐藏。

上面这种方法相当于 hack 了表单中的域值，大多数情况下，你不会想让占位文本随着表单一起发送给服务器端。只有启用了 JavaScript，用于 hack 占位文本的代码才会生效，因此我们可以使用 JavaScript 检测表单提交，并去掉那些与占位文本相匹配的值。代码第 66 行，我们捕获了表单提交事件，并清空了输入域中与占位文本相同的值。

jQuery 插件

开发人员可以通过编写插件的方式扩展 jQuery 库。添加自定义方法到 jQuery 函数上，插件即可与 jQuery 间无缝整合，其他开发人员引用了你的库后就能使用它了。下面是一个非常简单的示例，它的作用是弹出 JavaScript 的警告框：

```
jQuery.fn.debug = function() {
  return this.each(function(){
    alert(this.html());
  });
```

如果你希望对于页面上每个段落都能弹出警告框，则可像下面这样调用插件：

```
$("p").debug();
```

jQuery 插件的目的是遍历一个 jQuery 对象集合，而它们返回的也是对象集合，以便将它们链在一起。例如，因为 debug 插件会返回 jQuery 对象集合，所以我们可以用 jQuery 的 css 方法改变段落文本的颜色，并仅写一行代码。

```
$("p").debug().css("color", "red");
```

本书中，创建浏览器兼容性解决方案的时候，我们会多次用到 jQuery 插件来帮助组织代码结构。更多与 jQuery 相关的文档参见 http://docs.jquery.com/Plugins/Authoring

所有逻辑均已封装在了插件中，如下面所示将其与表单绑定，我们即可在页面上调用它。

Download html5placeholdertext/index.html

```
$(function(){
  function hasPlaceholderSupport() {
    var i = document.createElement('input');
    return 'placeholder' in i;
  }

  if(!hasPlaceholderSupport()){
    $("#create_account").placeholder();
    //placeholder_fallback 结束

    $('input[autofocus=true]').focus();

  };
});
```

至此，我们有了一套非常合宜的解决方案，无论用户使用哪种浏览器，Web 应用中都可使用占位文本。

3.4　实例 6：基于 contenteditable 属性实现在位编辑

我们一直在探索如何简化用户与应用间的交互过程。有时候，我们希望网站用户能够直接编辑一些信息，而无需使用额外的表单，即用户可以在位编辑。实现在位编辑的传统方法是监测文本区域上的单击事件，并将文本区域用文本框进行替换。随后，这些域通过 Ajax 将表单域中的已编辑过的文本值传回服务器端。现在，使用 HTML5 的 contenteditable 属性可以自动完成数据输入部分的工作。为了保存编辑过的数据，我们仍然需要编写少量用于向服务器端发送数据的 JavaScript 代码，但毕竟再也不用创建和隐藏可来回切换的表单了。

AwesomeCo 的某个项目允许用户复查自己的账户信息页。页面上显示了其名字、所在城市、所在州、邮编和电子邮件地址。接下来，我们就在这个页面上添加在位编辑功能，实现后的最终效果如图 3-6 所示。

User information

Name	Hugh Mann
City	Anytown
State	OH
Postal Code	92110
Email	boss@awesomecompany.com

图 3-6　在位编辑变得更容易

小试身手之前，我想让你了解一件事情，创建易访问的 Web 应用时，依赖于 JavaScript 实现某功能，却不首先在服务器端实现同样的功能是不好的做法。示例中并未实现服务器端解决方案，因为其目的是重点说明 contenteditable 属性的功能，而且这一示例代码不会上线。所以要记住，无论何时，首先构建一个不依赖于 JavaScript 的解决方案，然后实现依赖于 JavaScript 脚本的版本，最后为两个版本编写自动化测试代码，以便变更其中某个版本（但不变更另一个）时更可能捕获到 Bug。

3.4.1　账户表单

HTML5 规范引入了 contenteditable 属性，它几乎可用于任何元素之上。只要添加这一属性，即可将其变成可编辑区域。

Download **html5_content_editable/show.html**

```
<h1>User information</h1>
<div id="status"></div>
<ul>
  <li>
    <b>Name</b>
    <span id="name" contenteditable="true">Hugh Mann</span>
```

```
    </li>
    <li>
      <b>City</b>
      <span id="city" contenteditable="true">Anytown</span>
    </li>
    <li>
      <b>State</b>
      <span id="state" contenteditable="true">OH</span>
    </li>
    <li>
      <b>Postal Code</b>
      <span id="postal_code" contenteditable="true">92110</span>
      </li>
    <li>
      <b>Email</b>
      <span id="email" contenteditable="true">boss@awesomecompany.com</span>
    </li>
  </ul>
```

我们还可通过一些 CSS 为代码添加样式。我们使用一些 CSS3 选择器来识别可编辑的区域，使得当用户悬停或选中它们的时候，相应区域的颜色会发生变化。

Download html5_content_editable/show.html

```
Line 1   ul{list-style:none;}
     -
     -   li{clear:both;}
     -
     5   li>b, li>span{
     -     display: block;
     -     float: left;
     -     width: 100px;
     -   }
    10
     -   li>span{
     -     width:500px;
     -     margin-left: 20px;
     -   }
    15
     -   li>span[contenteditable=true]:hover{
     -     background-color: #ffc;
     -   }
     -
    20   li>span[contenteditable=true]:focus{
     -     background-color: #ffa;
     -     border: 1px shaded #000;
     -   }
```

前端代码编写完了。用户现在能够轻而易举地编辑页面上的数据。接下来，编写用于保存数据的代码。

3.4.2 持久化数据

虽然用户可以修改数据，但刷新页面或者离开当前页，修改后的数据会随之丢失。我们需要通过某种方法将修改后的数据提交到后台，还是借助 jQuery 来实现吧。如果以前使用过 Ajax，那么你不会对下面的代码感到陌生。

Download html5_content_editable/show.html

```
$(function(){
    var status = $("#status");
    $("span[contenteditable=true]").blur(function(){
      var field = $(this).attr("id");
      var value = $(this).text();
      $.post("http://localhost:4567/users/1",
         field + "=" + value,
         function(data){
           status.text(data);
         }
      );
    });

});
```

我们为每个 contenteditable 属性值为 true 的 span 标签添加事件监听器。剩下的任务就是提交数据到服务器端脚本。

3.4.3 回退

我们做了一堆事儿，但仍有一些受众无法享受我们提供的服务。首先，将编辑结果保存回服务器端依赖于 JavaScript，这不是一件好事。其次，当聚焦于文本区域时，我们使用了 focus 伪类来高亮显示相应区域，而某些版本的 IE 不支持 focus 伪类。下面，我们优先处理功能性的问题，然后再解决显示效果问题。

3.4.4 创批建编辑页面

依赖于 JavaScript 提交表单可能会将某些用户挡在门外，而与过分考虑多种解决方案不同，我们应该为他们提供备选方案，让他们去往一个独立的表单页面。当然，工作量变大了，但是考虑以下场景，你会觉得值了。

- 使用 IE7 的用户禁用了 JavaScript。
- 用户使用了不支持 HTML5 的浏览器。

- 用户使用了支持 HTML5 的最新版本的 Firefox，但还是禁用了 JavaScript，因为他不喜欢 JavaScript。（这种场景总是有的，而且出现的频率比你想象得要高。）

届时，最好的办法就是创建一个表单，用 POST 方式将表单内容赋给之前 Ajax 代码更新的目标。如何实现则取决于你自己，而很多框架都允许你检测请求类型，它们会查看 `accept` 头部信息以检测是常规的 POST 请求，还是 `XMLHttpRequest` 请求。这样一来，就能保证服务器端代码的 DRY 标准[1]。如果浏览器支持 `contenteditable` 属性和 JavaScript，我们会隐藏用于跳转到表单页面的链接。

因此，我们创建名为 edit.html 的新页面，编写标准的编辑表单，其 POST 内容将发送至原先 Ajax 版本的代码所更新的目标。

Download **html5_content_editable/edit.html**

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Editing Profile</title>
    <link href="style.css" rel="stylesheet" media="screen">
  </head>
  <body>
    <form action="/users/1" method="post" accept-charset="utf-8">
      <fieldset id="your_information">
        <legend>Your Information</legend>
        <ol>
         <li>
           <label for="name">Your Name</label>
           <input type="text" name="name" value="" id="name">
         </li>
         <li>
           <label for="city">City</label>
           <input type="text" name="city" value="" id="city">
         </li>
         <li>
           <label for="state">State</label>
           <input type="text" name="state" value="" id="state">
         </li>
         <li>
           <label for="postal_code">Postal Code</label>
           <input type="text" name="postal_code" value="" id="postal_code">
         </li>
         <li>
           <label for="email">Email</label>
```

① DRY 标准即“Don’t Repeat Yourself”，由 Dave Thomas 和 Andy Hunt 在《程序员修炼之道》[HT00]一书中首次提出。

```
            <input type="email" name="email" value="" id="email">
          </li>
        </ol>

      </fieldset>

      <p><input type="submit" value="Save"></p>
    </form>

  </body>
</html>
```

在 show.html 页面上添加一个链接：

Download html5_content_editable/show.html

```
<h1>User information</h1>
<section id="edit_profile_link">
  <p><a href="edit.html">Edit Your Profile</a></p>
</section>
<div id="status"></div>
```

添加链接后，仅需要对脚本进行小小地修改。当且仅当支持在位编辑的情况下，我们会在隐藏到编辑页面的链接，并启用 Ajax 支持。

Download html5_content_editable/show.html

```
if(document.getElementById("edit_profile_link").contentEditable != null){
```

下面是检测浏览器是否支持 `contenteditable` 属性的代码。

Download html5_content_editable/show.html

```
$(function(){
  if(document.getElementById("edit_profile_link").contentEditable != null){
    $("#edit_profile_link").hide();
    var status = $("#status");
    $("span[contenteditable=true]").blur(function(){
      var field = $(this).attr("id");
      var value = $(this).text();
      $.post("http://localhost:4567/users/1",
         field + "=" + value,
         function(data){
           status.text(data);
         }
      );
    });
  }

});
```

在此之后，我们的用户既可以使用标准界面进行编辑，也可以使用快速在位编辑模式。现在，你已经知道如何实现标准界面了，记住，一定要优先实现通用方案。与其他的兼容解决方案不同，如果没有实现这一兼容解决方案，部分功能在特定场景下将无法使用。

3.4.5 未来展望

到此为止，如果你在网站中添加了基于 JavaScript 的日期选择器，用户必须了解其用法。如果有过在线购买机票和预定旅馆房间的经历，那么你一定见过不同网站实现的各不相同的定制表单控件。类似于使用 ATM 机，不同机器的界面都有很大不同，会延缓你的操作速度。

想象一下，如果每个网站的日期选择都使用 HTML5 的 `date` 类型，并由浏览器来创建展现界面，那么用户无论访问哪个网站都能看到完全相同的日期选择器。屏幕阅读软件甚至能够实现一套标准机制来帮助盲人输入日期。再考虑一种场景，如果到处都是占位文本和自动聚焦功能，那么对于用户将多么有用。占位文本能够帮助屏幕阅读器向用户解释表单域的用途，而自动聚焦功能则可帮助用户在不用鼠标的情况下实现表单域上的导航，它不仅仅可为盲人提供便捷，也适用于那些因某种运动障碍而无法使用鼠标的用户。

开发人员可以将任意元素转换成可编辑区域，这使得实现在位编辑成了一件容易的事儿，但会潜在地改变我们为内容管理系统构建界面的方式。

如今的 Web 都是基于交互的，而表单是交互过程中至关重要的一环。我们能够使用 HTML5 提供的整套全新工具来提升用户体验。

第 4 章

用 CSS3 打造更好的用户界面

早在很久以前，开发人员就在代码中使用 CSS 来实现所需的效果。他们曾利用 JavaScript 或服务器端代码生成条纹样式的表格，或聚焦表单并对表单施加模糊效果。为了识别出要为 50 个文本框中的哪些添加样式，开发人员不得不大量使用带有额外 `class` 属性的标签。

但是以后不用这样了！CSS3 有一些非常神奇的页面元素选择器，有了它们，事情变简单了。或许你还记得，元素选择器是一种模式，它用于辅助查找 HTML 文档中的某些页面元素，以便找出元素后为其添加样式。我们将利用这些新的页面元素选择器渲染表格。然后，再看下如何利用其他一些 CSS3 特性改进网站的 CSS 打印样式表，我们还会尝试将内容拆分成多列。

本章中，我们会介绍下面的 CSS 特性。

- `:nth-of-type` `[p:nth-of-type(2n+1){color: red;}]`——查找某一特定类型的所有 *n* 个元素。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- `:first-child` `[p:first-child{color:blue;}]`——查找第一个子元素。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- `:nth-child` `[p:nth-child(2n+1){color: red;}]`——向后查找一个指定的子元素。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- `:last-child` `[p:last-child{color:blue;}]`——查找最后一个子元素。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- `:nth-last-child` `[p:nth-last-child(2){color: red;}]`——向前查找一个指定的子元素。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- `:first-of-type` `[p:first-of-type{color:blue;}]`——查找特定类型的第一个元素。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- `:last-of-type` `[p:last-of-type{color:blue;}]`——查找特定类型的最后一个元素。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]

- 对列的支持[#content{ column-count: 2; column-gap: 20px;column-rule: 1px solid #ddccb5; }]——将内容区域分成多列。[C2、F3.5、S3、O9.5、IOS3、A2]
- :after [span.weight:after { content: "lbs"; color: #bbb; }]——与 content 一起使用，用于在指定元素的后面追加指定的内容。[C2、F3.5、S3、IE8、O9.5、IOS3、A2]
- 媒体查询[media="only all and (max-width: 480)"]——基于设备设置应用样式。[C3、F3.5、S4、IE9、O10.1、IOS3、A2]

4.1 实例 7：使用伪类渲染表格

CSS 中的伪类（pesudoclass）是一种通过文档外信息或通过常规元素选择器无法表达的信息查找页面元素的方法。你可能已经使用过伪类，比如使用:hover 来改变鼠标悬停在超链接上时链接的颜色。使用 CSS3 中新的伪类可以更轻松地定位页面元素。

4.1.1 优化付款清单样式

AwesomeCo 用第三方清算记账系统来管理运输的物品。AwesomeCo 最大的市场之一就是会议用品，如笔、茶杯、衬衫及其他任何能印上公司标识的东西。客户要求付款清单具有更强的可读性，因此开发人员制作了如图 4-1 所示的标准 HTML 表格。

Item	Price	Quantity	Total
Coffee mug	$10.00	5	$50.00
Polo shirt	$20.00	5	$100.00
Red stapler	$9.00	4	$36.00
Subtotal			$186.00
Shipping			$12.00
Total Due			$198.00

图 4-1 当前付款清单仅使用了未加入样式的 HTML 表格

这是一张极为标准的付款清单，上面标有本次订单涉及的价格、数量、总计行，小计、运输总费用和合计总价。如果对相邻行使用不同的颜色显示，则会增加可读性而为合计总价使用其他颜色会更加突出其内容。

表格的代码如下。可以将下面的代码复制到你的文件中来运行。

Download css3advancedselectors/table.html

```
<table >
```

```
  <tr>
    <th>Item</th>
    <th>Price</th>
    <th>Quantity</th>
    <th>Total</th>
  </tr>
  <tr>
    <td>Coffee mug</td>
    <td>$10.00</td>
    <td>5</td>
    <td>$50.00</td>
  </tr>
  <tr>
    <td>Polo shirt</td>
    <td>$20.00</td>
    <td>5</td>
    <td>$100.00</td>
  </tr>
  <tr>
    <td>Red stapler</td>
    <td>$9.00</td>
    <td>4</td>
    <td>$36.00</td>
  </tr>
  <tr>
    <td colspan="3">Subtotal</td>
    <td>$186.00</td>
  </tr>
  <tr>
    <td colspan="3">Shipping</td>
    <td>$12.00</td>
  </tr>
  <tr>
    <td colspan="3">Total Due</td>
    <td>$198.00</td>
  </tr>
</table>
```

首先，让我们去掉难看的默认表格边框：

Download css3advancedselectors/table.css

```
table{
  width: 600px;
  border-collapse: collapse;
}

th, td{
  border: none;
}
```

然后，为表头略微添加修饰，用黑色填充背景并使用白色字体：

Download css3advancedselectors/table.css

```
th{
  background-color: #000;
  color: #fff;
}
```

应用样式后，得到如图 4-2 所示的表格。

Item	Price	Quantity	Total
Coffee mug	$10.00	5	$50.00
Polo shirt	$20.00	5	$100.00
Red stapler	$9.00	4	$36.00
Subtotal			$186.00
Shipping			$12.00
Total Due			$198.00

图 4-2

简单清理了表格边框并调整间距之后，开始应用伪类来渲染行和列。先从条纹化表格开始。

4.1.2 使用:nth-of-type条纹化表格的行

我们都见过“斑马纹”样式的表格。斑马纹很重要，因为它为用户提供了可参照的水平线。这种效果最好在表现层用 CSS 来实现。传统的做法是为表格的行添加额外的“odd”（奇）和“even”（偶）类名，并分别为“odd”和“even”类定义样式。我们不想在表格中引入额外的样式类，因为 HTML5 规范建议避免那些用于定义外观的类名。借助于新的选择器，我们可以轻易地实现上述功能而不必改变标记结构，进而彻底实现表现层与内容的分离。

nth-of-type 选择器可以查找某个特定类型中的第 *n* 个页面元素，这可以通过使用公式或关键字实现。因为关键字查找更容易理解，所以我们将首先介绍，而片刻之后，我们再来详细介绍如何用公式查找页面元素。

我们希望条纹化表格中的行，使相邻的两行颜色不同，那么最简单的方法就是找到所有偶数行，然后赋予其一种背景颜色。同样，我们也可以对奇数行赋予同一种颜色。CSS3 提供了 even 和 odd 两个关键字来支持这一特定的场景。

Download css3advancedselectors/table.css

```
tr:nth-of-type(even){
  background-color: #F3F3F3;
}
tr:nth-of-type(odd) {
```

```
  background-color:#ddd;
}
```

代码中选择器实现的是找到所有偶数行，对其着色，然后找到所有奇数行，用另一种颜色对其着色。这样就实现了斑马条纹，不需要额外的脚本代码，也不必在每行上加入额外的类名。

应用上面的样式后，表格如图 4-3 所示。

Item	Price	Quantity	Total
Coffee mug	$10.00	5	$50.00
Polo shirt	$20.00	5	$100.00
Red stapler	$9.00	4	$36.00
Subtotal			$186.00
Shipping			$12.00
Total Due			$198.00

图　4-3

下面，我们来改变表格列的对齐方式。

4.1.3　使用:nth-child对齐列文本

默认情况下，表格中所有列文本都是左对齐。接下来，我们要让除第一列外的所有列文本都右对齐。价格列和数量列右对齐后会更加清晰，可增强可读性。我们可以使用 nth-child 来实现，但是，首先得学习一下如何使用它。

nth-child 选择器用于查找某元素的子元素，与 nth-of-type 类似，我们可以使用关键字或公式。

公式就是 an+b，b 是偏移量，a 是倍数。没有上下文的情况下实在是不好理解其含义，所以我们在表格上下文中来试一下。

如果我们想选择所有行，可以这样使用选择器：

```
table tr:nth-child(n)
```

这里，我们未使用任何倍数，也未使用任何偏移量。

然而，如果我们想选择除了第一行（表格表头）之外的所有行，可以使用偏移量来实现：

```
table tr:nth-child(n+2)
```

如果我们想跳跃选择一些行，则可以使用倍数，如每两行选择一行用 2n：

```
table tr:nth-child(2n)
```

如果每 3 行选择一行则要用 3n。

同样，我们可以使用偏移量来改变起始行。下面这一选择器将从表格的第四行开始隔一行选择一行：

```
table tr:nth-child(2n+4)
```

这样一来，我们可以改变除第一列之外的其他列的文本对齐方式：

Download css3advancedselectors/table.css

```
td:nth-child(n+2){
  text-align: right;
}
```

此时，表格样式得到了真正的改善，如图 4-4 所示。

Item	Price	Quantity	Total
Coffee mug	$10.00	5	$50.00
Polo shirt	$20.00	5	$100.00
Red stapler	$9.00	4	$36.00
Subtotal			$186.00
Shipping			$12.00
Total Due			$198.00

图 4-4

我们再来渲染表格的最后一行。

4.1.4 使用:last-child加粗最后一行

现在，付款清单看起来已经很漂亮了，但是某位经理希望表格的最后一行能加粗显示以使此行更加突出。我们可以使用 last-child 来实现，它用于获取一组元素中的最后一个子元素。

Web 开发人员通常会为段落设置底边距以使页面看起来错落有致。不过，这种做法会导致一组元素的最后出现多余的底边距。例如，若段落位于侧边栏或标注框中，去掉最后一个段落的底边距能够减少最后一段与容器边缘之间的空间浪费。在此场景下，last-child 选择器是完美的选择，我们可以利用它来删除最后一段的底边距：

```
p{ margin-bottom: 20px }
#sidebar p:last-child{ margin-bottom: 0; }
```

我们再来用同样的方法来加粗表格的最后一行：

Download css3advancedselectors/table.css

```
tr:last-child{
  font-weight: bolder;
}
```

再来加粗表格的最后一列，突出显示合计总价列：

Download css3advancedselectors/table.css

```
td:last-child{
  font-weight: bolder;
}
```

最后，我们还要把汇总价格的字体变大一些。借助 `last-child` 选择器，我们可以找到表格中最后一个单元格，为其添加下面的样式（如图 4-5 所示）。

Download css3advancedselectors/table.css

```
tr:last-child td:last-child{
  font-size:24px;
}
```

Item	Price	Quantity	Total
Coffee mug	$10.00	5	**$50.00**
Polo shirt	$20.00	5	**$100.00**
Red stapler	$9.00	4	**$36.00**
Subtotal			**$186.00**
Shipping			**$12.00**
Total Due			**$198.00**

图　4-5

临近收工，但对于表格的后 3 行，还有几件事要做。

4.1.5　使用 :nth-last-child 向前查找元素

运费有折扣的时候，我们希望能够高亮显示“shipping”行。可以使用 `nth-last-child` 选择器快速定位此行。在 4.1.3 节中，我们演示了如何使用 `nth-child` 和 an+b 公式查找指定的子元素。`nth-last-child` 选择器的用法与 `nth-child` 选择器的用法几乎完全一样，唯一不同之处在于 `nth-last-child` 从最后一个元素开始反方向往前查找。正因为如此，用它来查找集合中倒数第二个元素非常简单。也就是说，只需找到付款清单表格的倒数第二行即可。

为了找到 shipping 行，我们使用了下面的代码：

Download css3advancedselectors/table.css

```
tr:nth-last-child(2){
  color: green;
}
```

上述代码中，我们定位到了倒数第二个子元素。

我们还需对表格做最后一次改进。之前，我们将除第一列以外的所有列都改为了右对齐，尽管修改后的条目描述列和价格列的行看起来还不错，但是表格的最后 3 行看起来有点不太协调。接下来，我们把表格最后 3 行的内容设为右对齐。借助 `nth-last-child` 选择器的公式，我们可以实现这个功能，方法是为公式中的 `n` 取负值并为 `a` 取正值，代码如下：

Download css3advancedselectors/table.css

```
tr:nth-last-child(-n+3) td{
  text-align: right;
}
```

可以把它理解为一个范围选择器。我们使用了 `nth-last-child`，使它偏移 3，意味着选择偏移元素之前的所有元素。如果使用 `nth-child`，则这个公式会选择从偏移开始的后面所有元素。

应用了新样式的表格如图 4-6 所示，看起来好多了，而且我们没有改动任何基础代码。实现中用到的诸多元素选择器尚无法在 IE 下使用，因此我们需要为此找到解决方案。

Item	Price	Quantity	Total
Coffee mug	$10.00	5	**$50.00**
Polo shirt	$20.00	5	**$100.00**
Red stapler	$9.00	4	**$36.00**
		Subtotal	**$186.00**
		Shipping	**$12.00**
		Total Due	**$198.00**

图 4-6　添加样式后的表格，完全使用 CSS3 实现的斑马纹和对齐方式

4.1.6　回退

最新版本的 Opera、Firefox、Safari 和 Chrome 版本都能识别这些元素选择器，但是 IE 8.0 及之前的版本会完全忽略它们。为此，你需要准备一套合适的兼容解决方案。

4.1.7　修改html代码

对于浏览器不兼容的问题，最常见且处处有效的解决方案就是修改底层代码。我们可以为表中各个单元格都赋予类样式，然后对每个类样式进行 CSS 定义。这恐怕是最糟糕的解决方案了，

它会导致表现层和内容完全混在一起，而之所以使用 CSS3 也恰恰是为了尽量避免这样的问题。某天不再需要额外的标记时，去掉它也将是一项痛苦的工作。

4.1.8 使用JavaScript

jQuery 库能够识别我们用到的大部分 CSS3 元素选择器，因此快速编写 jQuery 方法也能为表格添加样式。但是，还有更简单的方法。

Keith Clark 写了一个很棒而且简洁的库 IE-CSS3[①]，使 IE 浏览器可以支持 CSS3 的元素选择器了。我们需要做的只是在页面中添加一些脚本。

IE-CSS3 库可以使用 jQuery、Prototype 或者其他库，但是我更喜欢用 DOMAssistant 库[②]，因为这个库对此处使用的伪类支持得最好。

下载这些库，然后链接它们到你的文件中。因为此兼容方案仅针对 IE，因此，可以把相关逻辑都放在一个条件分支中，使其只在用户使用 IE 浏览器时生效。

Download css3advancedselectors/table.html

```
<!--[if (gte IE 5.5)&(lte IE 8)]>
  <script type="text/javascript"
          src="js/DOMAssistantCompressed-2.8.js"></script>
  <script type="text/javascript"
          src="js/ie-css3.js"></script>
<![endif]-->
```

将代码放在页面中即可实现对 IE 的兼容，效果可以参考图 4-7。

Table

Item	Price	Quantity	Total
Coffee mug	$10.00	5	$50.00
Polo shirt	$20.00	5	$100.00
Red stapler	$9.00	4	$36.00
		Subtotal	$186.00
		Shipping	$12.00
		Total Due	$198.00

图 4-7 IE 浏览器中展现的表格

只有开启浏览器对 JavaScript 的支持，表格的样式才会有助于内容的可读性。不过，即使没有样式的渲染，也不会妨碍用户查看清单的内容，只是不美观。

① 参见 http://www.keithclark.co.uk/labs/ie-css3/。
② 参见 http://www.domassistant.com/。

用 CSS3 渲染页面元素非常简便，尤其是当我们无法改变 HTML 结构时。无需添加额外的标记，仅利用语义层和新的元素选择器即可渲染页面元素，你会发现代码变得更易于维护了。

4.2 实例 8：使用:after 和 content 支持打印页面上的链接

CSS 不仅可以渲染已经存在的元素，还可以向文档中注入内容。某些情况下需要使用 CSS 实现内容生成，最典型的情况莫过于在用户打印网页时，在链接文本描述的后面追加超链接 URL 地址。在屏幕上浏览文档时，当鼠标悬停在链接上时，状态栏会显示链接指向哪个页面。但是，当我们查看网页的打印输出时，却完全看不出链接指向哪里。

AwesomeCo 正在开发一个关于服务条款的新页面，重构团队中的一名成员每次都要将整个网站打印下来。他希望能够清楚地知道每一个链接都指向哪里，以便决定是否做出调整。只需要一点 CSS 代码，就可以满足他的愿望，而且可以使之在 IE8、Firefox、Safari 和 Chrome 浏览器下正常工作。至于 IE6 和 IE7，我们得略施 JavaScript 小计。

现在，页面上除了一排链接啥也没有。最后，我们会将其放入模板中。

Download css3_print_links/index.html

```
    <ul>
      <li>
        <a href="travel/index.html">Travel Authorization Form</a>
      </li>
      <li>
        <a href="travel/expenses.html">Travel Reimbursement Form</a>
      </li>
      <li>
        <a href="travel/guidelines.html">Travel Guidelines</a>
      </li>
    </ul>

</body>
```

如果查看页面的打印输出结果，你完全不会知道链接都指向哪里，接下来让我们搞定这个问题。

4.2.1 使用CSS

为页面添加样式表时，我们可指定样式所适用的媒体类型。多数情况下，我们使用 `screen` 类型。然而，我们可选择 `print` 类型来定义样式表的应用范围，使其仅在页面打印时（或者打印预览时）加载生效。

Download css3_print_links/index.html

```
<link rel="stylesheet" href="print.css" type="text/css" media="print">
```

接下来创建依循如下规则的 print.css 文件：

Download css3_print_links/print.css

```
a:after {
  content: " (" attr(href) ") ";
}
```

这一样式对页面上的所有链接都生效，会在每个链接描述文字的后面加上括号中 href 属性值。使用时新浏览器打印页面时，打印结果如图 4-8 所示。

Forms and Policies

- Travel Authorization Form (travel/index.html)
- Travel Reimbursement Form (travel/expenses.html)
- Travel Guidelines (travel/guidelines.html)

图　4-8

如果仅想看到打印后的效果而不实际在纸上打印出来，可以使用浏览器的打印预览功能，打印预览操作同样会触发这一打印样式表。

示例中的样式会在除 IE6 和 ID7 以外的浏览器上生效，下面，让我们来解决 IE6 和 IE7 的问题吧。

4.2.2　回退

IE 浏览器支持一对 JavaScript 事件 onbeforeprint 和 onafterprint，我希望所有浏览器都可以支持它们。基于它们，我们可以在打印操作被触发时修改超链接的文本，并在打印结束时，将超链接的文本改回之前的内容。用户是不会注意到有什么不同的。[①]

我们只需创建 print.js 文件，编写如下代码：

Download css3_print_links/print.js

```
Line 1  $(function() {
     -    if (window.onbeforeprint !== undefined) {
     -        window.onbeforeprint = ShowLinks;
     -        window.onafterprint = HideLinks;
```

① 关于此项技术的概述可访问 http://beckelman.net/post/2009/02/16/Use-jQuery-to-Show-a-Linke28099s-Address-After-its-Text-When-Printing-In-IE6-and-IE7.aspx。

```
5     }
-   });
-
-   function ShowLinks() {
-     $("a").each(function() {
10      $(this).data("linkText", $(this).text());
-       $(this).append(" (" + $(this).attr("href") + ")");
-     });
-   }
-
15  function HideLinks() {
-     $("a").each(function() {
-       $(this).text($(this).data("linkText"));
-     });
-   }
```

接着，在页面中引用 print.js 文件。因为引用它仅是为了兼容 IE6 和 IE7，所以我们把相应逻辑放在条件分支中。这一代码需要 jQuery 的支持，为此，页面还需要应用 jQuery 库。

Download css3_print_links/index.html

```
  <script
    charset="utf-8"
    src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'
    type='text/javascript'>
  </script>
  <!--[if lte IE 7]>
  <script type="text/javascript" src="print.js"></script>
  <![endif]-->
</head>
<body>
  <h1>Forms and Policies</h1>

    <ul>
      <li>
         <a href="travel/index.html">Travel Authorization Form</a>
      </li>
      <li>
         <a href="travel/expenses.html">Travel Reimbursement Form</a>
      </li>
  <li>
     <a href="travel/guidelines.html">Travel Guidelines</a>
  </li>
</ul>
```

引用 print.js 文件后，我们就可以在各种浏览器下打印页面上的链接 URL 了。我们可以对打印样式文件进行修改，使其仅应用于网站中的某些链接，而不再对页面上的所有链接生效。

4.3　实例 9：创建多列布局

印刷行业使用多列排版已经好多年了，网页设计师也非常青睐这种排版模式。窄栏更适于读者阅读，随着显示区域变得越来越宽，开发人员也一直在找寻合适的列宽。毕竟，与在整页报纸上一行一行的阅读相比，谁也不愿意在显示器上从左到右的查看多行文本。在过去的十年中，出现过一些巧妙的解决方案来应对此问题，但是没有一个解决方案可以像 CSS3 规范提供的方法那样简单易懂。

4.3.1　分栏

AwesomeCo 每个月都会出版一份简报发给员工。公司使用了流行的 Web 电子邮件系统。基于电子邮件的简报并不漂亮而且难于维护。公司曾决定将简报放在内网的网站上，并计划通过电子邮件将简报的链接发送给所有员工。简报的页面原型如图 4-9 所示。



图 4-9　单列的简报太宽了，难以阅读

宣传部门的新主管有印刷行业的工作背景，她要求简报看起来更像真正的报纸，使之分两栏而不是一栏显示内容。

如果你尝试过用 `div` 加浮动的方式实现文本多栏显示，就会理解这种做法的难处。其最大的问题在于需要手工确定拆分文本的位置。在排版软件（如 InDesign）中，我们可以将文本框“链接”起来，以便当一个文本框被内容填满的时候，文本会进入与之相“链接”的文本区域。目前，Web 领域尚无类似的实现机制，但有其他简单易用的方法。我们可以取某个页面元素，拆分其内容为多列，使每一列具有相同的宽度。

编写简报所需的标记都是相当基础的 HTML 代码。因为简报内容会在写好后变动，所以在简报还没有正式发布前，先用占位文本来代替其真实内容。你可能会疑惑为什么我们不使用 section 等新的 HTML5 标签来实现，这是因为在 IE 浏览器中，我们的兼容方案无法与之兼容。

Download css3columns/condensed_newsletter.html

```
<body>
  <div id="header">
    <h1>AwesomeCo Newsletter</h1>
    <p>Volume 3, Issue 12</p>
  </div>
  <div id="newsletter">
    <div id="director_news">
      <div>
        <h2>News From The Director</h2>
      </div>
      <div>
        <p>
          Lorem ipsum dolor...
        </p>
        <div class="callout">
          <h4>Being Awesome</h4>
          <p>
            &quot;Lorem ipsum dolor sit amet...&quot;
          </p>
        </div>
        <p>
          Duis aute irure...
        </p>
      </div>
    </div>

    <div id="awesome_bits">
      <div>
        <h2>Quick Bits of Awesome</h2>
      </div>
      <div>
        <p>
          Lorem ipsum...
        </p>
        <p>
          Duis aute irure...
        </p>
      </div>
    </div>

    <div id="birthdays">
      <div>
        <h2>Birthdays</h2>
```

```
        </div>
        <div>
          <p>
            Lorem ipsum dolor...
          </p>
          <p>
            Duis aute irure...
          </p>
        </div>
      </div>

    </div>
    <div id="footer">
      <h6>Send newsworthy things to
        <a href="mailto:news@aweseomco.com">news@awesomeco.com</a>.
      </h6>
    </div>
  </body>
```

改为两列显示仅需要添加如下代码到样式表中：

Download css3columns/newsletter.html

```
#newsletter{
  -moz-column-count: 2;
  -webkit-column-count: 2;
  -moz-column-gap: 20px;
  -webkit-column-gap: 20px;
  -moz-column-rule: 1px solid #ddccb5;
  -webkit-column-rule: 1px solid #ddccb5;
}
```

图 4-10 所示的效果看起来更好了。我们可以为其加入更多的内容，浏览器会自动判断如何均匀地分列折行。注意，浮动的元素仍然浮动在列旁边而不是包含在列里。

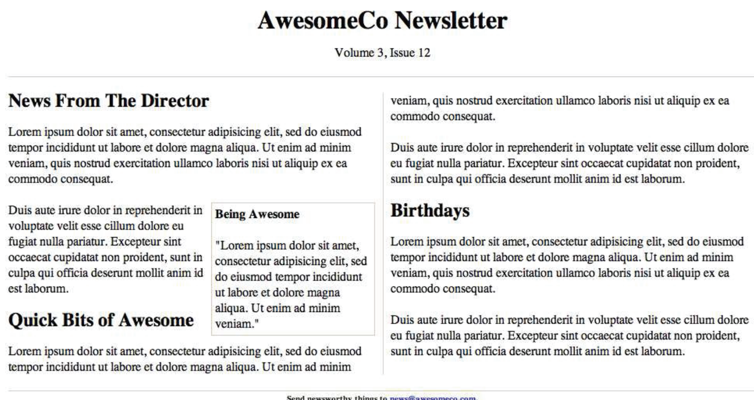
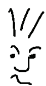


图 4-10　两列的简报



小乔爱问……

可以为各列设定不同的宽度吗？

不行。每列宽度必须相同。起初，我也觉得有些奇怪，所以在写这本书的时候，我对着规范又确认了一次，确实没有地方可以设置多种列宽。

不过，对比分栏的传统用法来看，这就不足为奇了。与表格不同，列栏的作用本不是实现简化版的网站侧边栏。分栏显示是为了让读者在阅读大篇幅的文章时更容易一些，而等宽列是最合适不过的选择。

4.3.2　回退

IE8 及其以下版本不支持 CSS3 的分栏显示，所以我们使用 jQuery 的 Columnizer 插件[①]来实现兼容方案。借助 Columnizer 插件，我们可以把内容均匀地拆分为多列，代码如下：

Download css3columns/newsletter.html

```
$("#newsletter").columnize({ columns: 2 });
```

没有开启 JavaScript 支持的用户只能看到单列显示的文本内容，但他一样可以阅读内容，因为与显示内容相关的标签是以线性方式组织的，所以代码适用于这种场景。我们可以通过 JavaScript 检测浏览器是否支持特定元素。如果检索已存在的 CSS 属性，则返回布尔值 `true`[②]。如果返回值是 `null`，则说明相应的 CSS 属性不可用。

Download css3columns/newsletter.html

```
<script
  charset="utf-8"
  src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'
  type='text/javascript'>
</script>

<script
  charset="utf-8"
  src="javascripts/autocolumn.js"
  type='text/javascript'>
</script>
```

① 参见 http://welcome.totheinter.net/columnizer-jquery-plugin/。

② 原文此处为“返回空字符串”，但根据代码，返回值应为布尔值 `true`。—— 译者注

```
<script type="text/javascript">
  function hasColumnSupport(){
    var element = document.documentElement;
    var style = element.style;
    if (style){
      return typeof style.columnCount == "string" ||
        typeof style.MozColumnCount == "string" ||
        typeof style.WebkitColumnCount == "string" ||
        typeof style.KhtmlColumnCount == "string";
    }
    return null;
  }

  $(function(){
    if(!hasColumnSupport()){
     $("#newsletter").columnize({ columns: 2 });
    }
  });
</script>
```

检查浏览器是否支持分栏显示，如果不支持，则使用插件。

刷新一下 IE 浏览器，可以看到两栏简报的效果。如图 4-11 所示，页面看起来也许还不完美，所以需使用 CSS 或 JavaScript 调整看上去不协调的元素，这些问题我留给你来处理。参考 4.1.8 节中的条件语句，我们能够针对特定版本的 IE 浏览器做特殊处理。

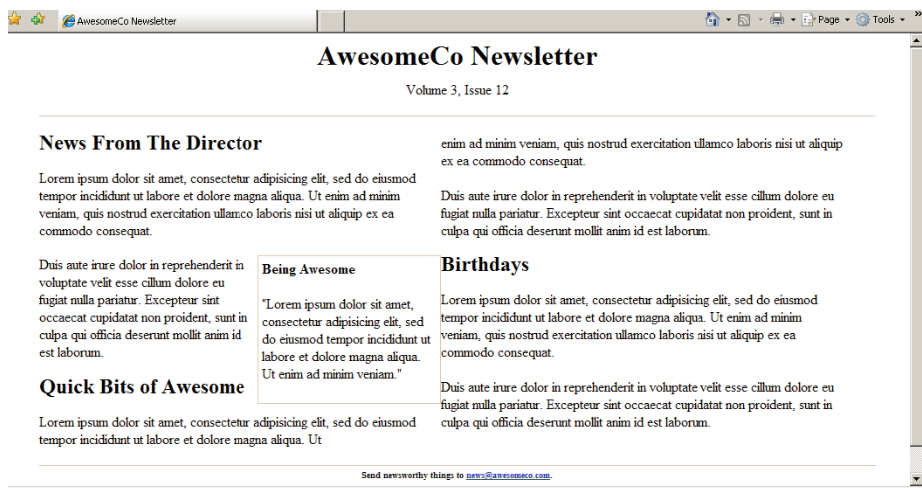


图 4-11　IE 浏览器的效果，但尚需进行细微调整

把文章分为多栏显示会提高文章的易读性。不过，文章较长时，用户可能不喜欢每次还要拖回最上方才能阅读下一栏。因此，请小心使用分栏。

4.4 实例 10：使用媒体查询构建移动设备界面

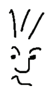
很早以前，我们就可以针对特定媒体定义其样式表，但正如在 4.2 节所讲述的那样，定义打印样式表时，输出的类型会受到限制。CSS3 的媒体查询[①]允许开发人员根据访客所用屏幕的尺寸来调整页面的渲染效果。一直以来，Web 开发人员使用 JavaScript 探测用户屏幕尺寸，进而获取与用户屏幕尺寸相关的信息。现在，同样的行为仅需设置样式表即可。媒体查询能够检测如下内容：

- 分辨率；
- 方向（纵向或者横向模式）；
- 设备的宽度和高度；
- 浏览器窗口的宽度和高度。

基于上面的描述，使用媒体查询可以更轻松地创建出适用于移动用户的可切换样式表。

AwesomeCo 的员工大多弃黑莓手机转投了 iPhone。对于 2.1 节的内容，市场部主管会乐于看到博客模板的 iPhone 版本，而其实现起来很容易。

目前，博客是两列布局，有一个主内容区域和一个侧边栏。增加博客页面在 iPhone 上可读性的最简单方式是移除浮动的元素，使侧边栏“掉落到”主内容区域的下方。这样一来，读者就不必在设备上向侧面滚动内容区了。



小乔爱问……

什么是手持媒体类型？

手持媒体类型的设计初衷是让我们可以像操纵打印机那样操纵移动设备，只不过大部分移动设备希望展现“真实的网络”，因此，它们会忽略媒体类型，取而代之的是将输出样式表与 screen 媒体类型相关联。

为了实现上述功能，需要在博客样式表的最后加入以下代码：

Download **css3mediaquery/style.css**

```
@media only screen and (max-device-width: 480px) {
  body{
    width:460px;
  }
```

① 参见 http://www.w3.org/TR/css3-mediaqueries/。

```
    section#sidebar, section#posts{
      float: none;
      width: 100%;
    }
  }
```

你可以认为媒体查询大括号里的代码是其自身的样式表，当查询条件满足时，就会被触发。示例中，我们重新定义了页面 body 元素的样式，并且将原来的两列显示改为了单列显示。

关联样式表时，同样可以使用媒体查询，我们将移动设备的样式表置于单独的文件中，代码如下：

```
<link rel="stylesheet" type="text/css"
  href="CSS/mobile.css" media="only screen and (max-device-width: 480px)">
```

现在，博客在 iPhone 上的可读性大大地提高了。使用媒体查询的方式来为其他显示设备（比如自助式服务一体机、平板电脑和其他任意尺寸的显示设备）制作样式表，让你的内容在更多设备上具有良好的可读性。

4.4.1 回退

Firefox、Chrome、Safari、Opera 和 IE 9 都支持媒体查询。你需要依赖于 JavaScript 兼容解决方案来应对不兼容的情况，基于用户的设备加载额外的样式表。此处示例是针对 iPhone 的，所以我们不需要兼容方案，因为即使不用媒体查询，内容依然可读。

不过，如果你想看一下媒体查询在其他浏览器上的效果，有一个 jQuery 插件[①]可供选择，它为其他浏览器提供了对基本媒体查询的支持。这一插件仅支持引用外部样式表的方式，而且仅支持以像素为单位的 min-width 和 max-width。尽管存在诸多的限制，对于窗口尺寸不一的界面来说也非常好用。

4.4.2 未来展望

本章中，我们讨论了用于改进用户界面的内容，但如果客户端的浏览器不支持这些新特性，用户还是能够照常工作。例如，表格没有条纹时，用户仍然可以看到表格的内容，页面元素没有圆角时，表单也仍有效，简报没有以多栏显示时，用户仍然可以阅读它。我们可在表现层解决上述效果问题，而不需要借助于 JavaScript 或服务器端的解决方案，这是一件好事。

目前，除了 IE 以外，几乎所有的浏览器都支持文中介绍的元素选择器。随着技术的普及，将来 IE 也会添加对其的支持，尤其是对伪类的支持。当规范变成终稿时，如 moz 和 webkit-等特定的前缀也会随之消失。那时，你就可以去掉实现兼容方案的代码了。

---

① 参见 http://plugins.jquery.com/project/MediaQueries。

第 5 章

增强可访问性

HTML5 中的很多新元素都可以帮助我们更准确地描述内容。在第三方程序解析我们的代码时，这点显得尤为重要。例如，有人使用屏幕阅读软件将屏幕上的图像内容翻译成文本并且大声朗读出来。屏幕阅读软件通过解析屏幕上的文本和标记来识别其中的链接、图片和其他元素。尽管屏幕阅读软件已经有了令人称奇的进步，但仍然赶不上当今趋势。一方面，页面上的一些动态区域往往难以检测，像轮询或者 Ajax 请求等都会改变页面上的内容。另一方面，对于更复杂的页面来说，由于屏幕阅读器需要朗读大量内容，所以语音浏览也很难做到。

富因特网应用的可访问性（WIA-ARIA[①]）规范提供了若干方法来提高网站尤其是 Web 应用的可访问性。对于在开发中使用了 JavaScript 和 Ajax 的应用来说，它会非常有用。WIA-ARIA 规范的一部分内容已经被纳入到了 HTML5 规范中，剩下的部分仍然独立并形成对 HTML5 规范的补充。许多屏幕阅读软件已经开始使用 WIA-ARIA 规范的功能了，比如 JAWS、WindowEyes，甚至 Apple 的内置 VoiceOver 功能。WIA-ARIA 还引入了额外的标记，为辅助技术提示页面上的可更新区域。

本章，我们将看到如何使用 HTML5 来提升使用辅助装置的残障用户的访问体验。最重要的是，本章介绍的技术不需要实现兼容性支持，因为现在多数的屏幕阅读软件都已采用了这些技术。

其中部分技术如下所示。

- role 属性[<div role="document">]——向屏幕阅读器说明某元素的用途。[C3、F3.6、S4、IE8、O9.6]
- aria-live [<div aria-live="polite">]——标识一个可能会被 Ajax 等技术自动更新的区域。[F3.6 (Windows)、S4、IE8]

---

① 参见 http://www.w3.org/WAI/intro/aria.php。

- aria-atomic [<div aria-live="polite" aria-atomic="true">]——标识某动态区域的阅读方式，即阅读全部内容还是只阅读发生了改变的内容。[F3.6 (Windows)、S4、IE8]

5.1 实例 11：使用 ARIA 角色提供导航提示

大部分网站的基本结构都类似，包括一个头部、一个导航区域、一些页面主要内容和一个尾部。这些网站就像批量生产出来的一样，代码也基本相同。不过弊端也显现出来了，这意味着屏幕阅读器必须按照特定顺序朗读网站的内容。由于大部分网站的不同页面都有相同的头部和导航，用户访问不同页面时必须听完每个页面上这些相同的内容。

推荐的解决方案是，提供一个隐藏的“跳过导航”链接，让屏幕阅读器朗读出来。此链接指向靠近主内容区域的一个锚。不过此功能并非内置的，而且不是所有开发人员都懂得（或者记得住）它的实现方式。

借助 HTML5 的新属性 role，开发人员可以设置各个元素在页面上的职责。屏幕阅读器可以轻松的解析页面并归类页面上所有元素的职责，这样就可以为页面建立一个简单的索引。例如，可以找到页面上所有 navigation 角色的元素，并把它们提供给用户，方便用户在应用程序中快速导航。

这些角色属性定义出自 WIA-ARIA 规范[1]，现已经被纳入了 HTML5 规范。有两种特有的角色我们现在就可以用：标志角色（landmark）和文档角色（document）。

5.1.1 标志角色

标志角色用来标识我们网站上“有意义的区域”，如 banner、搜索区，或导航等可以快速被屏幕阅读器识别的元素。

角　色	作　用
banner	识别页面中的banner区域
search	识别页面中的搜索区域
navigation	识别页面中的导航元素
main	识别页面中主要内容的起始处
contentinfo	识别内容的相关信息的位置，比如版权信息和发布日期
complementary	识别页面上作为主要内容的补充信息但是有自己独立意义的内容
application	识别一块包含Web应用的页面区域，而非Web文档

① 参见 http://www.w3.org/WAI/PF/aria/roles。

我们可以在 2.1 节内容的基础上，在 AwesomeCo 博客模板中应用上述多种角色。

在页面的总头部中，如下所示加入 banner 角色：

Download html5_aria/blog/index.html

```
<header id="page_header" role="banner">
  <h1>AwesomeCo Blog!</h1>
</header>
```

我们仅需要在已经存在的 header 标签中加入额外的 role= "banner"属性即可。

以同样方式，我们可以加入识别导航的角色：

Download html5_aria/blog/index.html

```
<nav role="navigation">
  <ul>
    <li><a href="/">Latest Posts</a></li>
    <li><a href="/archives">Archives</a></li>
    <li><a href="/contributors">Contributors</a></li>
    <li><a href="/contact">Contact Us</a></li>
  </ul>
</nav>
```

HTML5 规范规定了一些元素的默认角色，而且不能被改写。nav 元素一定有角色 navigation，严格来说无需被再次声明。虽然屏幕阅读器对此类默认规定的接受程度还不高，但是大部分屏幕阅读器可以理解这些 ARIA 角色。

主体区域和侧边栏可以定义如下：

Download html5_aria/blog/index.html

```
<section id="posts" role="main">
</section>
```

Download html5_aria/blog/index.html

```
<section id="sidebar" role="complementary">

  <nav>
    <h3>Archives</h3>
    <ul>
      <li><a href="2010/10">October 2010</a></li>
      <li><a href="2010/09">September 2010</a></li>
      <li><a href="2010/08">August 2010</a></li>
      <li><a href="2010/07">July 2010</a></li>
      <li><a href="2010/06">June 2010</a></li>
      <li><a href="2010/05">May 2010</a></li>
      <li><a href="2010/04">April 2010</a></li>
      <li><a href="2010/03">March 2010</a></li>
```

```
    <li><a href="2010/02">February 2010</a></li>
    <li><a href="2010/01">January 2010</a></li>
  </ul>
</nav>
</section> <!-- sidebar -->
```

对于在尾部定义的发布和版权信息可以使用 contentinfo 角色，如下：

Download html5_aria/blog/index.html

```
<footer id="page_footer" role="contentinfo">
  <p>&copy; 2010 AwesomeCo.</p>
</footer> <!-- footer -->
```

如果博客中含有搜索框，则同样可以为其赋予角色。现在，标志已经定义完了，接下来我们继续完善，让一些文档元素也可以被标识。

5.1.2 文档结构角色

借助文档结构（document structure）角色可以让屏幕阅读器轻松识别一些固定的内容，以便建立更利于导航的页面布局。

角　色	作　用
document	识别文档内容区域，相对于应用内容区域来说
article	识别组成文档独立内容的段落
definition	识别对短语或主题的定义
directory	识别对于一个集合元素的引用列表，比如一个内容表格。用于静态内容
heading	标识一个页面区域的头部
img	标识一段包含图片的区域。该区域包含图片或者标题和说明文字
list	标识一组不可交互的列表项
listitem	标识一个或者一组不可交互的列表项
math	标识一个数学表达式
note	标识对于主要内容的注释或补充
presentation	标识可以被辅助装置忽略的用于美化外观的内容
row	标识表格中的一行
rowheader	标识表格中包含字段含义的表格头

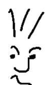
很多 document 角色可以被 HTML 标签明确定义，比如 article 和 header。然而，也有些角色（比如 document）不能被 HTML 标签明确定义，但是却非常有用，尤其当应用程序中动态内容和静态内容混合时。例如，一个基于 Web 的电子邮件客户端程序中，包含邮件内容的元素上可能含有 document 属性。这非常有用，因为屏幕阅读器通常使用键盘来做各种各样的导航。如果屏幕阅读器的关注点在应用程序元素上，可能需要通过键盘来浏览整个 Web 应用。然而，

如果屏幕阅读器的关注点在静态内容上，就可以换一种方式，通过屏幕阅读器的键绑定来实现。

我们可以在博客的 body 元素上添加 document 角色：

Download html5_aria/blog/index.html

```
<body role="document">
```



小乔爱问……

当我们可使用 nav 或者 header 元素时，是否还需要标志角色？

乍一看，标志角色似乎是多余的。但是当我们无法使用新元素的时候，标志角色却可以提供很大的灵活性。

使用 search 角色，可以提供给用户的不仅仅是搜索框，而且还会包含到网站地图的链接、一个快速导航的下拉列表，或者其他所有可以帮助用户快速获取信息的元素，而如果没有 search 角色，则提供给用户的就只是一个搜索框。

相对于新元素和表单控件，规范中引入的角色要更多。

这将使屏幕阅读器将该页面当做静态内容来处理。

5.1.3 回退

由于结合最新版本的浏览器和最新版本的屏幕阅读器已经可以支持这些角色，所有我们可以放心使用这些角色了。不为其提供支持的浏览器会简单地忽略它们，因此使用这些角色只会帮助可以利用他们的人。

5.2 实例 12：创建可访问的可更新区域

为解决对 Web 应用中 Ajax 的识别，我们已经做了很多尝试。标准的做法是通过改变页面上已更新内容的视觉效果来提示用户某些变动已经产生了。然而，一位使用屏幕阅读器的用户是无法看到这些视觉效果的。WIA-ARIA 规范提供了一个完美的替代解决方案，目前多款知名的屏幕阅读软件都可以在 IE、Firefox 和 Safari 中使用该功能。

AwesomeCo 通讯主管想要一个新的主页，新主页上需要有通往 services（服务）部分、contact（联系信息）部分、about（关于我们）部分的链接。他要求主页不能有滚动条，因为“人们讨厌滚动”。他希望我们能做个页面原型，将页面上的菜单链接水平放置，当单击菜单链接时，改变页面的主要内容。这个主页实现起来非常简单。使用 aria-live 属性，我们可以做一些以前无法

做好的事情，即使用支持屏幕阅读器的方法实现此种界面。

让我们实现一个简单的页面，如图 5-1 所示。我们在主页上放置了所有内容，如果可以使用 JavaScript，我们将隐藏除了第一条记录之外的所有内容。我们将使用页面锚使导航链接指向各个部分。我们将使用 jQuery 把页面锚的链接替换为事件，使主要内容被换出。使用 JavaScript 的用户可以看到我们的主管想要的效果，禁用了 JavaScript 的用户也可以看到页面上的所有内容。



图 5-1 使用 jQuery 改变页面主要内容的主页原型

5.2.1 创建页面

从创建一个基本的 HMTL5 页面开始，作为用户刚进入主页看到的默认内容，我们为其加上 welcome 部分（欢迎页面）。下面是含有导航条和跳转链接的页面代码：

Download html5_aria/homepage/index.html

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>AwesomeCo</title>
    <link rel="stylesheet" href="style.css" type="text/css">
  </head>
  <body>
    <header id="header">
      <h1>AwesomeCo </h1>
      <nav>
        <ul>
          <li><a href="#welcome">Welcome</a></li>
          <li><a href="#services">Services</a></li>
          <li><a href="#contact">Contact</a></li>
          <li><a href="#about">About</a></li>
        </ul>
      </nav>
    </header>
    <section id="content"
             role="document" aria-live="assertive" aria-atomic="true">
```

```
    <section id="welcome">
      <header>
        <h2>Welcome</h2>
      </header>
      <p>The welcome section</p>
    </section>
  </section>
  <footer id="footer">
    <p>&copy; 2010 AwesomeCo.</p>
    <nav>
      <ul>
        <li><a href="http://awesomeco.com/">Home</a></li>
        <li><a href="about">About</a></li>
        <li><a href="terms.html">Terms of Service</a></li>
        <li><a href="privacy.html">Privacy</a></li>
      </ul>
    </nav>
  </footer>

</body>
</html>
```

welcome 部分有一个 ID——`welcome`，与导航条中的锚对应。我们可以通过同样的方式声明页面中的其他部分。

Download html5_aria/homepage/index.html

```
<section id="services">
  <header>
    <h2>Services</h2>
  </header>
  <p>The services section</p>
</section>
<section id="contact">
  <header>
    <h2>Contact</h2>
  </header>
  <p>The contact section</p>
</section>

<section id="about">
  <header>
    <h2>About</h2>
  </header>
  <p>The about section</p>
</section>
```

上述 4 个内容区域都放在如下标记中：

Download html5_aria/homepage/index.html

```
<section id="content"
         role="document" aria-live="assertive" aria-atomic="true">
```

折行代码内的属性告诉屏幕阅读器该区域会被动态更新。

5.2.2 polite和assertive更新

使用 aria-live 来提醒用户页面上的变化有两种方法。polite 模式不打断用户的阅读。举例来说，如果一个用户的屏幕阅读器正在朗读一个句子，同时另外一部分页面的内容发生了更新。那么如果是设置为 polite 模式，则屏幕阅读器会等当前句子阅读完毕后再阅读更新后的内容，而模式设置为 assertive 的话，则将把段落更新当做更高优先级的事情，屏幕阅读器会立刻停止阅读当前内容，开始阅读新内容。在构建网站时，慎重选择这两种模式非常重要。过度使用 assertive 会使人困惑。因此，请仅在必要时使用 assertive 模式。在我们的例子中，是应该使用它的，因为我们将把其他内容都隐藏起来。

5.2.3 atomic更新

第二个属性（aria-atomic=true）告知屏幕阅读器朗读已变化区域的全部内容，如果我们将其设置为 false，则告知屏幕阅读器仅仅朗读已变化的节点内容。由于我们是替换区域内的所有内容，因此在我们的例子中，告诉屏幕阅读器阅读所有内容是合理的。如果我们要使用 Ajax 替换一个列表项或者附加一个表格条目，则应使用 false。

5.2.4 隐藏区域

为了隐藏一些区域，需要为页面写一些 JavaScript 代码。我们将创建一个名为 application.js 的文件，然后连同 jQuery 库一起放到我们的页面中。

Download html5_aria/homepage/index.html

```
<script type="text/javascript"
   charset="utf-8"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>

<script type="text/javascript"
   charset="utf-8"
   src="javascripts/application.js">
</script>
```

application.js 文件中包含了如下简单的代码：

Download html5_aria/homepage/javascripts/application.js

```
Line 1  // 支持 IE6、IE7 和 IE8 的 HTML5 结构元素
     -  document.createElement("header");
     -  document.createElement("footer");
     -  document.createElement("section");
     5  document.createElement("aside");
     -  document.createElement("article");
     -  document.createElement("nav");
     -
     -  $(function(){
    10
     -    $("#services, #about, #contact").hide().addClass("hidden");
     -    $("#welcome").addClass("visible");
     -
     -    $("nav ul").click(function(event){
    15
     -      target = $(event.target);
     -      if(target.is("a")){
     -        event.preventDefault();
     -        if ( $(target.attr("href")).hasClass("hidden") ){
    20          $(".visible").removeClass("visible")
     -            .addClass("hidden")
     -            .hide();
     -          $(target.attr("href"))
     -             .removeClass("hidden")
    25             .addClass("visible")
     -             .show();
     -        };
     -      };
     -
    30    });
     -
     -  });
```

第 11 行，我们隐藏了 services、about、contact 部分，还对它们应用了 hidden 类。然后再下一行，我们对默认的 welcome 区域应用了 visible 类。添加了这些类，我们在做切换的时候就很容易确定哪些区域需要关闭，以及何时需要切换。

在第 14 行，我们会捕捉对导航条的所有单击操作。在第 17 行，我们会判断单击的是什么元素。如果单击的是一个链接，我们将会判断相应部分是否为隐藏状态。通过 jQuery 的元素选择器，被单击链接的 href 属性可以轻松识别将要跳转到的目标区域，如第 19 行所示。

如果对应部分是隐藏状态，则需要将其他部分都置为隐藏状态，并将选择的部分设置为显示状态。这样就完成了，屏幕阅读器软件会检测到其内容的更新。

5.2.5 回退

像角色一样, 最新版本的屏幕阅读器现在就可以使用该功能了。通过尝试一些好的实践，比如使用简单的 JavaScript 代码，我们就可完成一个面向相当广泛受众的实现。老版本的浏览器和屏幕阅读器会忽略这些新增的附加属性，因此加入这些属性到我们的页面中不存在任何风险。

5.2.6 未来展望

HTML5 和 WIA-ARIA 规范为更具有可访问性的 Web 铺平了道路。有了可以识别页面中更新区域的能力，开发者可以开发丰富的 JavaScript 应用，而不用太过担心可访问性问题。

第 6 章

在 canvas 上绘图

之前我们讨论的是基础架构和界面设计，从本书的第二部分开始，我们将介绍如何使用 HTML5 和 CSS3 画图、操纵多媒体文件和制作个性化界面元素。先从使用 HTML5 的新元素 canvas 作图开始。

如果想在 Web 应用中加入图片，通常我们会先打开绘图软件创建一幅图片，然后使用 img 标签将其包含到 Web 页面中。如果想要创建动画，就要用到 Flash。借助 HTML5 的 canvas 元素，开发人员可以使用 JavaScript 在浏览器中以编程方式创建图片和动画。不论简单还是复杂的图形，都可以利用 canvas 创建出来，甚至在不使用服务器端库、Flash 或其他插件的情况下，也可以创建图形或图表。本章，我们将实现这些内容。

<canvas> [<canvas><p>Alternative content</p></canvas>]——支持通过 JavaScript 创建矢量图形。[C4、F3、IE9、S3.2、O10.1、IOS3.2、A2]

首先，我们先通过绘制简单的图形来了解如何使用 JavaScript 的 canvas 元素。比如做 AwesomeCo 的一个 logo。然后，我们使用 canvas 专用的图形库创作一个条形图，用于浏览器统计。我们还会讨论一些兼容性方面的挑战，这是未来必然会面对的问题，因为 canvas 不仅仅是页面元素更是编程接口。

6.1 实例 13：绘制 logo

作为一个容器元素，canvas 就像 script 元素一样，不过它是一块供我们在上面绘画的白板。我们可以定义一个带有长宽属性的 canvas，如下所示：

Download html5canvasgraph/canvas_simple_drawing.html

```
<canvas id="my_canvas" width="150" height="150">
  Fallback content here
</canvas>
```

遗憾的是，我们无法在维持内容不变的情况下，使用 CSS 控制或改变 canvas 元素的长宽。因此，需要在定义 canvas 时就指定 canvas 的尺寸。

我们可使用 JavaScript 将图形绘制在 canvas 上。即使我们对不支持 canvas 元素的浏览器提供了回退方案，也仍然需要阻止 JavaScript 代码试图操控它。通过 id 属性找到 canvas，检查浏览器是否支持 canvas 元素的 getContext 方法：

Download html5canvasgraph/canvas_simple_drawing.html

```
var canvas = document.getElementById('my_canvas');
if (canvas.getContext){
  var context = canvas.getContext('2d');

}else{
  // 显示用于替代 canvas 的内容
  // 或者让浏览器显示<canvas>元素里面的文本内容
}
```

如果 getContext 方法有返回结果，则从 canvas 中获取一个 2D 上下文，然后就可以向其中添加对象了。如果没有获取到上下文，则需要提供显示替代内容的回退方法。由于 canvas 元素需要使用 JavaScript 来进行控制，因此，我们一开始就要搭建一套回退框架。

获取到 canvas 的上下文后，就可以很方便地在其中添加元素了。添加一个红色的方块，需要先设置填充颜色，然后创建方块，如下所示：

Download html5canvasgraph/canvas_simple_drawing.html

```
context.fillStyle = "rgb(200,0,0)";
context.fillRect (10, 10, 100, 100);
```

canvas 的 2D 上下文是一个网格，默认情况下，左上角是坐标原点。当创建图形时，需要指定起始坐标 X、Y 和长宽（参见图 6-1）。

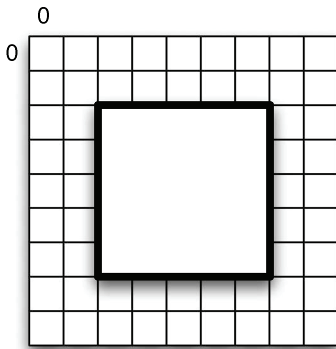


图　6-1

每个图形都会被添加到自己的层。因此，我们可以创建 3 个相互间 10 像素偏移的方块，如下所示：

Download html5canvasgraph/canvas_simple_drawing.html

```
context.fillStyle = "rgb(200,0,0)";
context.fillRect (10, 10, 100, 100);
context.fillStyle = "rgb(0,200,0)";
context.fillRect (20, 20, 100, 100);

context.fillStyle = "rgb(0,0,200)";
context.fillRect (30, 30, 100, 100);
```

它们会彼此堆叠起来，如图 6-2 所示。



图　6-2

在了解了基本的绘图操作之后，让我们看下 AwesomeCo 的 logo。这是一个非常简单的图片，如图 6-3 所示。



图 6-3　AwesomeCo 的 logo

6.1.1　绘制logo

该 logo 包含一行文字、一条折线、一个正方形和一个三角形。首先创建一个 HTML5 文件，并添加一个 canvas 元素，然后创建绘制 logo 的 JavaScript 函数，该函数会检测是否可以使用 2D 上下文。

Download html5canvasgraph/logo.html

```
var drawLogo = function(){
```

```
    var canvas = document.getElementById('logo');
    var context = canvas.getContext('2d');
};
```

我们在检查了 canvas 元素的存在与否之后，立即调用该方法，如下：

Download html5canvasgraph/logo.html

```
$(function(){
  var canvas = document.getElementById('logo');
  if (canvas.getContext){
    drawLogo();
  }
});
```

这里我们再次使用 jQuery 函数确保在文档准备就绪时能触发事件。我们要在页面上查找 ID 为 logo 的元素，因此，最好确保将 canvas 元素添加到文档中，这样才能找到它，才能使检测代码正常工作。

Download html5canvasgraph/logo.html

```
<canvas id="logo" width="900" height="80">
  <h1>AwesomeCo</h1>
</canvas>
```

接下来向 canvas 中添加文字“AwesomeCo”。

6.1.2 添加文字

向 canvas 中添加文字的过程包括选择字体、字号和设定对齐方式，然后添加文字到坐标网格中对应的坐标上。我们可以如下所示向 canvas 中添加文字“AwesomeCo”：

Download html5canvasgraph/logo.html

```
context.font = 'italic 40px sans-serif';
context.textBaseline = 'top';
context.fillText('AwesomeCo', 60, 0);
```

上面代码中，在应用文字到 canvas 之前，我们先定义了字体样式、文字的基线（即定义了垂直对齐方式）。然后我们使用 fillText 方法用填充色填充文字，设置右边距为 60 个像素，为后边要绘制的大三角形路径留出空间。

6.1.3 绘制线条

在 canvas 上划线，使用的是“点连接”的方法。先在 canvas 坐标网格上定义一个起始点，然后定义将要连接的其他点。当在 canvas 上移动时，将点连接成线，如图 6-4 所示。

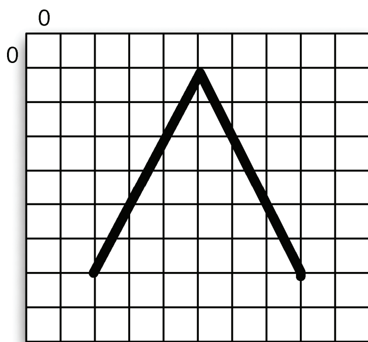


图 6-4

我们使用 beginPath()方法开始划线，然后创建路径，如下：

Download html5canvasgraph/logo.html

```
context.lineWidth = 2;
context.beginPath();
context.moveTo(0, 40);
context.lineTo(30, 0);
context.lineTo(60, 40 );
context.lineTo(285, 40 );
context.stroke();
context.closePath();
```

当完成在 canvas 上的移动之后，必须调用 stroke 函数将线画出来，然后调用 closePath 方法停止绘画动作。

现在剩下了大三角形里面的正方形和三角形组合成的图形。

6.1.4 移动原点

我们需要在大三角形里面画一个小的正方形和三角形。在绘制图形和路径时，可以指定相对于 canvas 左上角原点的 X 和 Y 坐标，也可以将原点移到新的位置。

我们将原点移动后再在里面绘制小正方形：

Download html5canvasgraph/logo.html

```
context.save();
context.translate(20,20);
context.fillRect(0,0,20,20);
```

需要注意的是，在移动原点之前要先调用 save()方法，保存 canvas 之前的状态以方便之后恢复。就像一个还原点，我们也可以将其理解为一个栈。每次调用 save()函数都会得到一个新

的记录。当我们完成绘画后，调用 restore()就可以恢复到栈最上部还原点的状态。

下面使用路径来绘制里面的三角形，我们将使用填充法来达到像是三角形嵌入了正方形的效果：

Download html5canvasgraph/logo.html

```
context.fillStyle    = '#fff';
context.strokeStyle = '#fff';

context.lineWidth = 2;
context.beginPath();
context.moveTo(0, 20);
context.lineTo(10, 0);
context.lineTo(20, 20 );
context.lineTo(0, 20 );
context.fill();
context.closePath();
context.restore();
```

我们设置了笔触，在开始绘画之前，将其填充为白色（#fff）。然后开始画线，因为之前移动了原点，所以现在是相对于正方形左上角来定位的。

即将大功告成，现在只剩下调色了。

6.1.5 添加颜色

在 6.1.4 节，我们简要地介绍了如何为绘制工具设置笔触和填充颜色。在绘制任何东西之前，添加如下代码就可以将绘制内容的颜色设置为红色：

Download html5canvasgraph/logo.html

```
context.fillStyle = "#f00";
context.strokeStyle = "#f00";
```

但是这么做有些单调，我们可以创建渐变对象并应用渐变到笔触和填充颜色中：

Download html5canvasgraph/logo_gradient.html

```
// context.fillStyle = "#f00";
// context.strokeStyle = "#f00";
var gradient = context.createLinearGradient(0, 0, 0, 40);
gradient.addColorStop(0,   '#a00'); // 暗红色
gradient.addColorStop(1,   '#f00'); // 红色
context.fillStyle = gradient;
context.strokeStyle = gradient;
```

我们仅需要创建一个渐变对象，设置渐变的停止颜色。在本例中，我们只是在两种色调的红色之间应用渐变。有必要的话，还可以画出彩虹。[①]

---

① 请勿滥用彩虹色！

需要注意的是，我们需要在画之前先设置颜色。

到此为止，logo 就完成了，在这个过程中，我们也更好地理解了如何在 canvas 上绘制简单的图形。然而，IE9 之前版本的浏览器是不支持 canvas 元素的，因此我们要解决这个兼容性问题。

6.1.6 回退

Google 发布了一个名为 ExplorerCanvas[①]的库，为 IE 用户开放了几乎所有的 Canvas API。我们需要做的仅仅是将该库包含进来：

Download html5canvasgraph/logo_gradient.html

```
<!--[if lte IE 8]>
<script src="javascripts/excanvas.js"></script>
<![endif]-->
```

这么做理论上能解决 IE 用户的问题，但是实际效果并没有那么好。在撰写本书时，最稳定的版本都还无法支持文字的添加，而从 Subversion 版本控制库[②]中获取的版本其字体是有问题的。而且，该库还不支持为笔触添加渐变。

因此，我们需要依赖其他解决方案，例如在 canvas 元素内放置一个 PNG 格式的 logo 图片，或者根本就不使用 canvas。本例仅仅是作为一个练习，来介绍如何在 canvas 上绘画，所以即使无法让其完美运行在跨平台的生产系统中，也不是什么大事。

6.2 实例 14：使用 RGraph 绘制统计图

AwesomeCo 网站的内容比较多，高级主管希望能看到网络状态的统计图。虽然可以让后台开发人员实时获取这些数据，但是他们更希望看到你是否有办法在浏览器中以图形的形式将其显示出来。为此，他们提供了一些测试数据。我们的目标是将测试数据转换为如图 6-5 所示的图形。

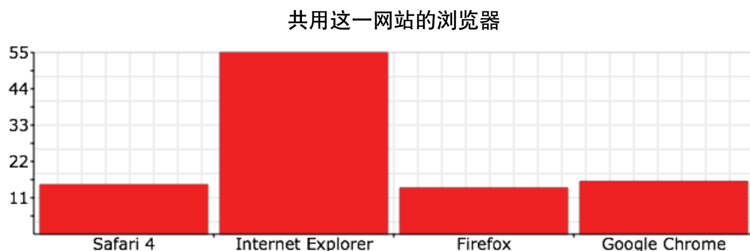


图 6-5 使用 canvas 生成的客户端条形图

---

① 参见 http://code.google.com/p/explorercanvas/。

② 参见 http://explorercanvas.googlecode.com/svn/trunk/excanvas.js。

在 Web 页面上画图有很多种方法。开发人员通常会选择用 Flash 来显示图形，然而在类似 iPad 和 iPhone 等移动设备上，Flash 会受到限制。也有一些服务器端的解决方案，但是如果处理实时数据就太费处理器了，而 canvas 是基于标准的一种客户端解决方案，是很好的选择，只要我们能小心处理，就能确保它在旧版本浏览器中也能正常工作。现在我们已经知道如何绘制正方形了，但是如果想绘制更复杂的图形，则需要更多的 JavaScript 代码。这需要借助图形库来实现。

虽然 HTML5 还没有大量普及，但这并没有影响到开发者开发 RGraph[①]的进程。有了 RGraph，在 HTML5 canvas 上画图就会变得异常简单。它是一个纯 JavaScript 解决方案，因此如果用户的客户端禁用了 JavaScript，它就无法工作了，继而 canvas 也就无法工作了。下面是绘制简单的条形图的一段代码：

Download html5canvasgraph/rgraph_bar_example.html

```
<canvas width="500" height="250" id="test">[no canvas support]</canvas>

<script type="text/javascript" charset="utf-8">
  var bar = new RGraph.Bar('test', [50,25,15,10]);
  bar.Set('chart.gutter', 50);
  bar.Set('chart.colors', ['red']);
  bar.Set('chart.title', "A bar graph of my favorite pies");
  bar.Set('chart.labels', ["Banana Creme", "Pumpkin", "Apple", "Cherry"]);
  bar.Draw();
</script>
```

我们所需要做的就是创建一系列 JavaScript 数组，然后库会自动在 canvas 上将图画出来。

6.2.1 使用HTML描述数据

我们可以在 JavaScript 代码中对浏览器的统计数据进行硬编码。但这样做的话，只有没禁用 JavaScrpit 的用户才能看到这些数据。另一种方法是把数据作为文本放在页面上。然后使用 JavaScript 读取这些数据，稍后将其提供给图形库。

Download html5canvasgraph/canvas_graph.html

```
<div id="graph_data">
  <h1>Browser share for this site</h1>
  <ul>
    <li>
      <p data-name="Safari 4" data-percent="15">
        Safari 4 - 15%
      </p>
    </li>
```

① 参见 http://www.rgraph.net/。

```
      <p data-name="Internet Explorer" data-percent="55">
        Internet Explorer - 55%
      </p>
    </li>
    <li>
      <p data-name="Firefox" data-percent="14">
        Firefox - 14%
      </p>
    </li>
    <li>
      <p data-name="Google Chrome" data-percent="16">
        Google Chrome - 16%
      </p>
    </li>
  </ul>
</div>
```

我们使用 HTML5 的数据属性来存储浏览器的名字和占比。尽管我们有这些信息的文本形式，但是通过编程处理这些数据更简单，因为不需要进行字符串解析。

通过浏览器打开该页面或者直接看图 6-6，可以看到图形数据被清晰显示出来并且即使没有图形也具有很强的可读性。这可以作为移动设备或者其他无法使用 canvas 或禁用了 JavaScript 的用户的替代内容。

共用这一网站的浏览器

- Safari 4 - 15%
- Internet Explorer - 55%
- Firefox - 14%
- Google Chrome - 16%

图 6-6　图形的 HTML 描述

接下来，我们将上述标记转换为图形。

6.2.2　将HTML内容转换为条形图

为显示条形图，我们需要使用 RGraph 的 Bar 图形库和 RGraph 核心库，还需要使用 jQuery 从文档中抓取数据。在 HTML 页面的 head 区域，需要将库包含进来。

Download html5canvasgraph/canvas_graph.html

```
<script type="text/javascript"
  charset="utf-8"
```

```
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>
<script src="javascripts/RGraph.common.js" ></script>
<script src="javascripts/RGraph.bar.js" ></script>
```

为了创建图形，我们需要从 HTML 文档中抓取图形的标题、标签（label）、数据，然后传递给 RGraph 库。RGraph 在数组中存放接受到的标签和数据。我们可以使用 jQuery 快速建立这些数组。

Download html5canvasgraph/canvas_graph.html

```
Line 1  function canvasGraph(){
   -      var title = $('#graph_data h1').text();
   -
   -      var labels = $("#graph_data>ul>li>p[data-name]").map(function(){
   5         return $(this).attr("data-name");
   -      });
   -
   -      var percents = $("#graph_data>ul>li>p[data-percent]").map(function(){
   -         return parseInt($(this).attr("data-percent"));
  10      });
   -
   -      var bar = new RGraph.Bar('browsers', percents);
   -      bar.Set('chart.gutter', 50);
   -      bar.Set('chart.colors', ['red']);
  15      bar.Set('chart.title', title);
   -      bar.Set('chart.labels', labels);
   -      bar.Draw();
   -
   -  }
```

首先，在第 2 行，我们获取了标题的文本信息。在第 4 行，我们选取了所有带有 data-name 属性的元素，使用 jQuery 的 map 方法将这些信息转换到了一个数组中。

第 8 行使用了同样的逻辑，获得了占比数据的数组。

得到了这些数据，RGraph 就可以轻松地画出图形了。

6.2.3 显示备用内容

在 6.2.1 节，我们本可以把图像放在 canvas 的开始和结束标签之间。这样就可以在浏览器支持 canvas 的时候隐藏这些元素，而在浏览器不支持 canvas 的时候将其显示出来。然而，当浏览器支持 canvas 但是用户禁用了 JavaScript 的时候仍然会隐藏它们。

jQuery CSS 和 CSS

在本章，我们在创建元素的时候使用 jQuery 为其添加属性信息。诸如标签的颜色和条形图的颜色等大部分属性信息，应该被写到一个独立的样式表中，特别是如果开发人员希望能够针对脚本独立改变的属性。对于原型，本例中的方法没有问题，但是如果作为生产版本的话，一定要记得分离表现、行为和内容。

我们也可以把数据放在 canvas 的外面，然后在检查到 canvas 存在的情况下，使用 jQuery 数据隐藏起来。

Download html5canvasgraph/canvas_graph.html

```
var canvas = document.getElementById('browsers');
if (canvas.getContext){
  $('#graph_data').hide();
  canvasGraph();
}
```

这样，图形就可以显示了，除非用户的浏览器不支持 canvas 元素。

6.2.4　回退

在构建这一解决方案时，我们已经解决了可访问性和禁用 JavaScript 时的回退问题，但是我们还可以为那些不支持 canvas 但是可以使用 JavaScript 的用户提供一幅备用图片。

其实图形库有很多，但是每个库抓取数据的方法各不相同。条形图其实就是一些具有特定高度的长方形，而且我们具备创建页面图形所需的所有数据。

Download html5canvasgraph/canvas_graph.html

```
Line 1  function divGraph(barColor, textColor, width, spacer, lblHeight){
     -    $('#graph_data ul').hide();
     -    var container = $("#graph_data");
     -
     5    container.css( {
     -      "display" : "block",
     -      "position" : "relative",
     -      "height": "300px"}
     -     );
    10
     -    $("#graph_data>ul>li>p").each(function(i){
     -
     -      var bar = $("<div>" + $(this).attr("data-percent") + "%</div>");
     -      var label = $("<div>" + $(this).attr("data-name") + "</div>");
```

```
15
-       var commonCSS = {
-                   "width": width + "px",
-                   "position" : "absolute",
-                   "left" : i * (width + spacer) + "px"};
20
-       var barCSS = {
-                "background-color" : barColor,
-                "color" : textColor,
-                "bottom" : lblHeight + "px",
25               "height" :  $(this).attr("data-percent") + "%"
-       };
-       var labelCSS = {"bottom" : "0", "text-align" : "center"};
-
-       bar.css( $.extend(barCSS, commonCSS) );
30      label.css( $.extend(labelCSS,commonCSS) );
-
-       container.append(bar);
-       container.append(label);
-     });
35
-   }
```

第 2 行隐藏了无序列表，这样就看不到文本信息了。然后抓取包含图形数据的元素，并对这些元素应用基本的 CSS 样式。第 6 行，我们将元素的位置属性设置为 relative，使条形图和标签信息在容器内保持绝对位置。

然后我们循环遍历无序列表中的段落（11 行），并创建条形。每循环遍历一次标签，就创建两个 div 元素，一个针对条形，一个针对标签信息（标签在条形的下面）。因此只需简单的计算和一些 jQuery 代码，我们就可以重建这一图形，虽然看起来不完全一致，但对于证明概念来说已经足够了。

接下来，我们只需将代码放在 canvas 中去测试，如下所示：

Download html5canvasgraph/canvas_graph.html

```
var canvas = document.getElementById('browsers');
if (canvas.getContext){
  $('#graph_data').hide();
  canvasGraph();
}
else{
  divGraph("#f00", "#fff", 140, 10, 20);
}
```

图 6-7 中为兼容版本。通过使用 JavaScript、HTML 和 CSS，我们创建了一个适用于多种平

台浏览器的客户端条形图和统计信息。使用 canvas 有一个额外的好处，它促使我们从一开始就考虑兼容性问题，而不是在后期修修补补。这对于可访问性非常有益。

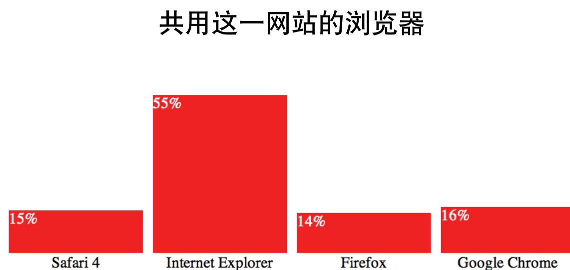
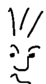


图 6-7 Internet Explorer 中的图形

这是创建图形最方便和最灵活的方法，可以轻松创建图形并附上相应的可供替代的文本信息。这样，大家就都可以分享重要的数据了。



小乔爱问……

为什么不试一下 ExplorerCanvas?

我们在 6.1.6 节提到过 ExplorerCanvas，它可以与 RGraph 很好地配合使用。RGraph 甚至发布了带有 ExplorerCanvas 的版本。然而，这个综合版本只适用于 Internet Explorer 8，如果用户使用 IE7 或更老的版本，就不得不提供类似于我们之前讨论过的回退方案。我建议大家留意一下 ExplorerCanvas，因为人们对该库的更新维护很积极。或者，你也可能想通过自己的技巧来解决这个问题。

6.2.5 未来展望

既然我们已经对 canvas 的工作原理略知一二，那接下来就可以想一下可以应用 canvas 的地方了。我们可以利用 canvas 来创建游戏、建立媒体播放器的用户界面，或者使用它绘制更有艺术性的图片，而你所需要的只是一些简单的 JavaScript 代码和一点点想象力。

现在，Flash 比 canvas 更有优势，因为对 Flash 的支持很广泛，但随着 HTML5 的普及，canvas 也会被广泛地应用，更多的开发人员会利用 canvas 在浏览器中创建简单的二维图形。canvas 不需要任何额外的插件，也比 Flash 耗费更少的 CPU，尤其是在 Linux 和 OS X 上。而且，canvas

提供了在 Flash 不可用时创建二维图形的机制。随着支持 canvas 的平台越来越多，其速度和功能也会更加完善，同时也会有越来越多的开发工具和库来帮助我们做更加神奇的事情。

不过 canvas 不会止步于二维图形，canvas 规范最后还会支持三维图形。浏览器厂商也正在实现硬件加速。对于仅仅使用 JavaScript 就能创建出有趣的用户界面和引人入胜的游戏，canvas 让这一切成为了可能。

第 7 章

嵌入音频和视频

在当今因特网上，音频和视频占据着极其重要的位置。尽管因特网上的播客、音频预览甚至视频教程比比皆是，但直到现在，也只能通过浏览器插件运行它们。对于如何在页面中嵌入音频和视频，HTML5 引入了新的方法。本章，我们不仅会讨论一些在页面中嵌入音频和视频的方法，还要确保使用旧浏览器的用户也能正常浏览它们。

本章我们要讨论的是下面两个元素。

- *<audio>* [<audio src="drums.mp3"></audio>]——浏览器原生播放音频。[C4、F3.6、IE9、S3.2、O10.1、IOS3、A2]
- *<video>* [<video src="tutorial.m4v"></video>]——浏览器原生播放视频。[C4、F3.6、IE9、S3.2、O10.5、IOS3、A2]

在讨论这两种元素之前，我们有必要先了解一下 Web 上音频和视频的发展历史。毕竟要弄明白我们要去哪儿，必须先知道我们在哪儿。

7.1 发展历史

很早以前开发人员就尝试在 Web 页面上使用音频和视频了。这种尝试始于人们在自己的主页中嵌入 MIDI 文件，当时使用 embed 标签来引用文件，比如：

```
<embed src="awesome.mp3" autostart="true"
   loop="true" controller="true"></embed>
```

因为 embed 标签一直无法成为标准，所以开发人员换用被纳入了 W3C 标准的 object 标签。为了兼容不支持 object 标签的旧浏览器，通常在 object 标签中还要嵌套 embed 标签，例如：

```
<object>
<param name="src" value="simpsons.mp3">
<param name="autoplay" value="false">
<param name="controller" value="true">
```

```
<embed src="awesome.mp3" autostart="false"
   loop="false" controller="true"></embed>
</object>
```

但是，并非所有的浏览器都能以这种方式显示内容，也不是所有的服务器都针对它进行了正确的配置。在 Web 上的视频开始变得越来越流行之后，情况就变得越来越复杂了。我们开始在 Web 上碰到各种各样的音频和视频，包括 RealPlayer、Windows Media，以及 QuickTime。每个厂商都有自家的视频标准，而且似乎每个网站用来编码视频的方法和格式都不相同。

Macromedia（已被 Adobe 收购）很早就意识到，在跨平台实现音频与视频的传送方面，它的 Flash Player 可能是最佳载体。当时近 97%的 Web 浏览器都已支持 Flash，当内容提供者发现他们只需要做一次编码即可在所有平台上播放其内容时，无数的网站开始转向使用 Flash 传播音频和视频。

Apple 特立独行，于 2007 年发布了 iPhone 和 iPod touch，同时决定不在这些设备上提供对 Flash 的支持。内容提供商随即作出响应，实现了视频在 Mobile Safari 浏览器中的正常播放。这些视频采用 H.264 编码，也能通过普通的 Flash Player 播放，这样内容提供商仍然可以一次编码而使内容可运行于多种平台。

HTML5 规范的编写者认为，浏览器应该原生支持音频和视频的播放，而不应该依赖于一堆 HTML 的插件。这就是 HTML5 音频和视频开始变得越来越有意义的地方，即将音频和视频作为 Web 内容中的“一等公民”。



小乔爱问……

Flash 已经可以跨浏览器使用了，那为什么不继续使用它呢？

最简单的回答是，作为一位开发人员，还没有一个浏览器厂商规定我们可以对嵌入页面中的内容进行什么操作。我们可以使用 CSS 和 JavaScript 操作元素，而不必像现在这样胡乱向 Flash 影片中传递参数。另外，随着标准越来越成熟，这种状况也会逐渐好转。

7.2 容器和编解码器

说起 Web 上的视频，从另一个角度来说其实讨论的是容器和编解码器。我们可以把数码相机中导出来的 AVI 或者 MPEG 文件看做是视频，但这样理解太过简单。容器就像是一个外壳，其中包含了音频流和视频流，甚至有时候还包含一些字幕之类的元数据。这些音频和视频流需要

进行编码，于是就出现了编解码器。视频和音频有成百上千种编码方式，但对于 HTML5 视频，仅需要一少部分。

7.2.1 视频编解码器

当我们观看视频的时候，视频播放器需要对视频进行解码。然而我们使用的播放器可能无法解码我们想看的视频。一些播放器使用软件来解码视频，比较慢或者会大量占用 CPU。有些播放器使用硬件解码，但却限制了可播放的视频格式。现在，要使用 HTML5 的 `video` 标签的话，我们需要了解 3 种视频格式：H.264、Theora 以及 VP8。

编码格式及其支持的浏览器

- H.264 [IE9、S4、C3、IOS]
- Theora [F3.5、C4、O10]
- VP8 [IE9 如果编解码器已安装、F4、C5、O10.7]

1. H.264

H.264 是一种高质量的编解码器标准，由 MPEG 组创建并在 2003 年标准化。为了在兼容诸如手机之类的低端设备的同时兼顾到高端设备的视频处理，H.264 规范分成了好几类。[①]通用属性在所有类中都有涵盖，但高端类中增加了一些可选的特性，用来提高视频质量。举例来说，iPhone 和 Flash Player 都支持 H.264 格式视频的播放，但 iPhone 只支持低质量的“baseline”类，而 Flash Player 则支持高质量视频流。我们可以一次将视频编码为不同的类，这样就能在不同的平台上实现平滑播放。

由于 Microsoft 和 Apple 都支持 H.264 编码，因此 H.264 已经成为事实标准。除此之外，Google 的 YouTube 将其视频编码转换为了 H.264 格式，以便在 iPhone 上播放，而且 Adobe 的 Flash Player 也对它提供支持。不过 H.264 不是开放的技术，它受专利保护，授权后方可使用。内容提供商必须在支付版税后才能使用 H.264 进行视频编码，不过对于免费提供给终端用户的内容来说，是无需支付版税的。[②]

自由软件的支持者们担心版权所有者最终会要求内容提供商支付高昂的版税。正是这种担心，促进了替代性编码器的开发和推广。

---

① H.264 编码体系定义了 4 种不同的 Profile（类）：Baseline（基线类）、Main（主要类）、Extended（扩展类）和 High Profile（高端类）。——译者注

② 参见 http://www.reelseo.com/mpeg-la-announces-avc-h264-free-license-lifetime/。

2. Theora

Theora是由Xiph.Org组织开发的免版权编码标准。对于内容提供商来说，通过Theora可以创建出与使用H.264时同样效果的视频，但是设备制造商采用此标准的步伐有点慢。不需要任何额外软件的辅助，Firefox、Chrome和Opera就能够在任意平台上播放Theora格式的视频，但是Internet Explorer、Safari和iOS设备却不行。Apple和Microsoft担心“潜水艇专利”。所谓“潜水艇专利”是指故意延迟专利的发布，在其他人实现这一技术时保持低调。时机成熟时，专利申请人“浮出水面”，对毫无戒备的市场收取专利使用费。

3. VP8

Google的VP8是一项完全开源、免版税的编码标准。并且用其创建的视频质量可同H.264视频相媲美。支持VP8的浏览器有Mozilla、Google Chrome和Opera。Microsoft承诺只要用户安装过任一款编解码器，其Internet Explorer 9就可以支持VP8。Adobe的Flash Player也同样支持VP8，因此VP8是非常值得关注的替代品。但是Safari和iOS设备不支持VP8，这就意味着尽管VP8是免费的，但是要想在iPhone或者iPad上面发布视频，那么内容提供商仍需使用H.264编解码器。

7.2.2 音频编解码器

看似相互竞争的不同视频标准方面的状况还不够复杂，但不同音频标准方面的竞争也需要我们关注。

编码格式及其支持的浏览器:

- AAC [S4、C3、IOS]
- MP3 [IE9、S4、C3、IOS]
- Vorbis (OGG) [F3、C4、O10]

1. 高级音频编码（AAC）

这是Apple在其iTunes Store中使用的音频格式。它的设计初衷是，在相同文件大小的情况下提供比MP3更好的音质。同H.264类似，AAC也提供多种音频类。它与H.264的另外一个共同点就是，AAC也不是一项免费的编码标准，有相应的授权费。

Apple的所有产品都支持AAC文件，而Adobe的Flash Player和开源的VLC播放器也支持它。

2. Vorbis（OGG）

这是一款开源的免版税格式，Firefox、Opera和Chrome都对其提供支持。我们可以发现OGG

也可以同 Theora 和 VP8 视频编解码器相配合。Vorbis 文件的音频质量非常好，但是对其提供支持的硬件音乐播放器却不多。

3. MP3

尽管 MP3 格式非常普遍和流行，但 Firefox 和 Opera 却不对其提供支持，因为它也受专利保护。Safari 和 Google Chrome 支持 MP3。

视频编解码器和音频编解码器需要打包在一起才能发布和播放。现在我们来看看视频容器。

7.2.3 容器和编解码器协同工作

容器是一个元数据文件，用于识别和混合音频或视频文件。实际上，容器中并不包含关于其中所含信息的编码方式的信息，关键是容器“包装”音频和视频流。通常容器可以保存任意类型的已编码媒体组合，不过我们将会在讨论在 Web 上处理视频时看到这些组合。

- OGG 容器，内含 Theora 视频和 Vorbis 音频，得到了 Firefox、Chrome 和 Opera 的支持。
- MP4 容器，内含 H.264 视频和 AAC 音频，Safari 和 Chrome 对其提供支持，同时可以被 Adobe 的 Flash Player 播放，也可以在 iPhone、iPod 和 iPad 上播放。
- WebM 容器，使用 VP8 视频和 Vorbis 音频，被 Firefox、Chrome、Opera 和 Adobe Flash Player 支持。

鉴于 Google 和 Mozilla 正在推进 VP8 和 WebM，所以 Theora 最终会走向消亡。但是从事物的外在来看，我们仍然需要对视频进行两次编码，即为 Apple 用户编码一次（尽管 Apple 的桌面份额很小，但在美国移动设备中所占的份额却非常大），然后为 Firefox 和 Opera 用户编码一次，因为这两个浏览器都拒绝支持 H.264。[①]

说来话长，不过现在我们已经了解了一些编解码器和容器的历史及限制，接下来让我们深入到实际实现中，先从音频开始。

7.3 实例 15：音频

AwesomeCo 正在开发一个网站，要在网站中展示截屏视频（screencast）中可以使用的免版税音频。应该会有一个示范页面，页面中展示一组独立的音频片段。完成之后，我们会看到一个音频片段列表，并且访问者可以快速试听每个音频片段。我们无需担心如何收集这个项目中的各类音频片段，因为客户端的音频工程师已经为我们准备好了 MP3 和 OGG 格式的音频样本。附录

① 参见 http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2009-June/020620.html。

C 给出了关于如何编码音频文件的一些信息。

7.3.1 建立基本列表

音频工程师已经为我们提供了 4 个音频样本：drums、organ、bass 和 guitar。我们需要使用 HTML 标记来分别描述它们。下面是对 drums 的描述：

Download html5_audio/audio.html

```
<article class="sample">
  <header><h2>Drums</h2></header>
  <audio id="drums" controls>
    <source src="sounds/ogg/drums.ogg" type="audio/ogg">
    <source src="sounds/mp3/drums.mp3" type="audio/mpeg">
    <a href="sounds/mp3/drums.mp3">Download drums.mp3</a>
  </audio>
</article>
```

我们首先定义了一个 audio 元素，并且指明需要显示一些控件。接下来为文件定义多个播放源。先定义音频样本的 MP3 和 OGG 版本，接着显示一个超链接，使得在浏览器不支持 audio 元素的情况下，用户可以通过这个超链接直接下载 MP3 文件。

这段非常简短的代码在 Chrome、Safari 和 Firefox 中都可以运行。我们继续将其他 3 份音频样本的代码一并添加到 HTML5 模板中：

Download html5_audio/audio.html

```
<article class="sample">
  <header><h2>Drums</h2></header>
  <audio id="drums" controls>
    <source src="sounds/ogg/drums.ogg" type="audio/ogg">
    <source src="sounds/mp3/drums.mp3" type="audio/mpeg">
    <a href="sounds/mp3/drums.mp3">Download drums.mp3</a>
  </audio>
</article>
<article class="sample">
  <header><h2>Guitar</h2></header>
  <audio id="guitar" controls>
    <source src="sounds/ogg/guitar.ogg" type="audio/ogg">
    <source src="sounds/mp3/guitar.mp3" type="audio/mpeg">
    <a href="sounds/mp3/guitar.mp3">Download guitar.mp3</a>
  </audio>
</article>

<article class="sample">
  <header><h2>Organ</h2></header>
  <audio id="organ" controls>
    <source src="sounds/ogg/organ.ogg" type="audio/ogg">
```

```
        <source src="sounds/mp3/organ.mp3" type="audio/mpeg">
        <a href="sounds/mp3/organ.mp3">Download organ.mp3</a>
      </audio>
    </article>

    <article class="sample">
      <header><h2>Bass</h2></header>
      <audio id="bass" controls>
        <source src="sounds/ogg/bass.ogg" type="audio/ogg">
        <source src="sounds/mp3/bass.mp3" type="audio/mpeg">
        <a href="sounds/mp3/bass.mp3">Download bass.mp3</a>
      </audio>
    </article>
  </body>
</html>
```

当我们在支持 HTML5 的浏览器中打开这个页面的时候，如图 7-1 所示，列表中的每个实体都会有自己的音频播放器。在这个页面中，按下 Play 按钮后，浏览器会自己处理音频的播放。

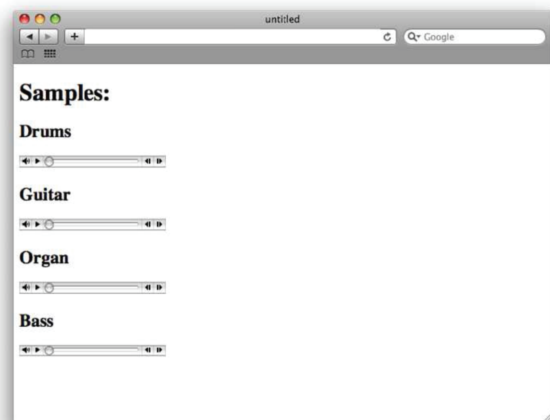


图 7-1　Safari 中的页面效果

在 Internet Explorer 中打开这个页面的话，我们会看到下载链接，因为浏览器不识别 audio 标签。这是一种尚佳的回退方案，不过还有更合适的，让我们来试试看。

7.3.2　回退

audio 元素自身内嵌了音频的回退支持。我们使用 source 元素定义了多种音频源，并且提供了音频文件的下载链接。如果浏览器无法解析 audio 元素，它就会显示我们放在里面的链接。我们还可以更进一步，在定义完音频源之后，使用 Flash 作为回退方案。

然而这也许并非最佳方案。因为我们很可能遇到一个支持 audio 元素，但不支持所提供音频格式的浏览器。例如，开发人员可能觉得无需浪费时间去为同一个音频准备多种格式的文件源。此外，HTML5 规范特别指出，audio 的回退支持内容不应是将被屏幕阅读器读取的。

最简单的方案是将下载链接移出 audio 元素，然后用 JavaScript 隐藏它：

Download html5_audio/advanced_audio.html

```
<article class="sample">
  <header><h2>Drums</h2></header>
  <audio id="drums" controls>
    <source src="sounds/ogg/drums.ogg" type="audio/ogg">
    <source src="sounds/mp3/drums.mp3" type="audio/mpeg">
  </audio>
  <a href="sounds/mp3/drums.mp3">Download drums.mp3</a>
</article>
```

然后我们只需要检测浏览器对 audio 的支持性，然后隐藏链接即可。具体的做法是，在 JavaScript 中创建一个新的 audio 元素，检测其对 canPlayType()方法的响应：

Download html5_audio/advanced_audio.html

```
var canPlayAudioFiles = !!(document.createElement('audio').canPlayType);
```

我们评测响应结果，然后把代码中所有嵌套的锚都隐藏掉：

Download html5_audio/advanced_audio.html

```
$(function(){
  var canPlayAudioFiles = !!(document.createElement('audio').canPlayType);

  if(canPlayAudioFiles){
    $(".sample a").hide();
  };
});
```

音频的回退方案相对简单，并且一些用户实际上还挺喜欢这样直接下载文件的功能。

浏览器原生播放音频只是抛砖引玉。浏览器中音频和视频的 HTML5 JavaScript API 才刚刚起步，详见 7.4 节的“JavaScript 的媒体内容 API”。

7.4　实例 16：嵌入视频

AwesomeCo 希望在其 Web 站点中展示其新制作的一系列视频教程，并且希望这些视频能支持尽量多的设备，特别是 iPad。作为一次实验，我们已经在“Photoshop Tips”系列中提供了两个视频，我们将会用它们建立原型。我们已经有了 H.264、Theora 和 VP8 格式的视频文件，因此可以专注于编写页面。[①]

① 了解更多关于编码视频文件的信息，参见附录 C。

video 标签同 audio 非常类似。我们只需要提供视频源，然后在无需任何插件的情况下，Chrome、Firefox、Safari、iPhone、iPad 以及 Internet Explorer 9 就能直接播放视频了。第一个视频文件 01_blur 的标记如下：

Download **html5video/index.html**

```
<article>
  <header>
    <h2>Saturate with Blur</h2>
  </header>
  <video controls>
    <source src="video/h264/01_blur.mp4">
    <source src="video/theora/01_blur.ogv">
    <source src="video/webm/01_blur.webm">
    <p>Your browser does not support the video tag.</p>
  </video>
</article>
```

我们定义的是带有控件的 video 标签。不使用 autoplay 属性的意思是暗示浏览器不要自动播放视频。

为确保 Web 浏览器知道如何处理视频文件，我们还需新建一个.htaccess 文件，并使之与为视频定义 MIME 类型的 Web 页面位于同一文件夹中，如下所示：

Download **html5video/.htaccess**

```
AddType video/ogg  .ogv
AddType video/mp4  .mp4
AddType video/webm .webm
```

一旦将这些文件上传到 Web 服务器，我们的视频就可在大多数浏览器中播放，用户将会看到如图 7-2 所示的视频播放器。

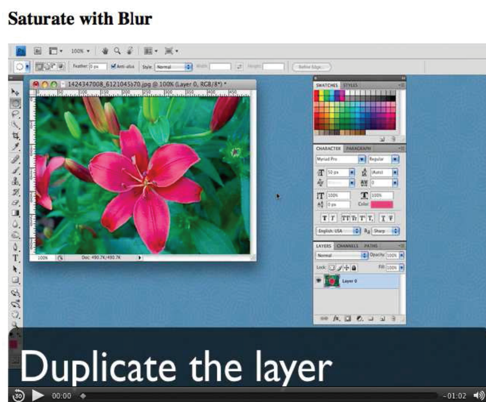


图 7-2 使用 Safari 的 HTML5 视频播放器播放视频

我们还无法兼顾使用 Internet Explorer 8 及更老浏览器的用户。对此的处理方式是使用 Flash。

7.4.1 回退

我们将 Flash 对象代码放置在 video 标签中，目的在于既能提供基于 Flash 的回退支持，同时又不影响 HTML5 video 的使用。Video For Everybody[1]网站中对此有详细的描述，不过这里我们只做基本的实现。

Flowplayer[2]是一款基于 Flash 的播放器，可以播放使用 H.264 编码的视频。下载该播放器的开源版本，然后将 flowplayer-x.x.x.swf 和 flowplayer-controls-x.x.x.swf 两个文件放置在项目的 swf 文件夹中，这样文件结构就组织好了。

接下来将下述代码写在 video 标签内，紧跟在最后一个 source 元素之后：

Download html5video/index.html

```
<object width="640" height="480" type="application/x-shockwave-flash"
   data="swf/flowplayer-3.2.2.swf">
   <param name="movie" value="swf/flowplayer-3.2.2.swf" />
   <param name="allowfullscreen" value="true" />
   <param name="flashvars"
     value='config={"clip":{"url":"../video/h264/01_blur.mp4",
                            "autoPlay":false,
                            "autoBuffering":true
                           }
                    }' />
   <img src="video/thumbs/01_blur.png"
     width="640" height="264" alt="Poster Image"
     title="No video playback capabilities." />
 </object>
```

特别要注意这段代码：

Download html5video/index.html

```
<param name="flashvars"
  value='config={"clip":{"url":"../video/h264/01_blur.mp4",
                         "autoPlay":false,
                         "autoBuffering":true
                        }
                 }' />
```

视频文件的路径应该是针对 Flowplayer 的相对路径。由于我们将 Flowplayer 放在了 swf 文件夹内，因此为了确保播放器能顺利播放视频，应该使用路径../video/h264/01_blur.mp4。

---

① 参见 http://videoforeverybody。
② 参见 http://flowplayer.org/download/index.html。

这里使用了自闭合标签。之前我们提到过，在 HTML5 中无需对<img>和<source>等空标签进行自闭合。但是有一些旧的浏览器不能正确识别<source>元素，进而无法显示回退内容。这种情况下，建议对<source>元素进行自闭合。

当我们在 Internet Explorer 中浏览此页面的时候，视频可以正常播放，无需再将其转换为其他格式，这多亏了 Flowplayer。使用 Internet Explorer 的用户将会看到如图 7-3 所示的页面。①

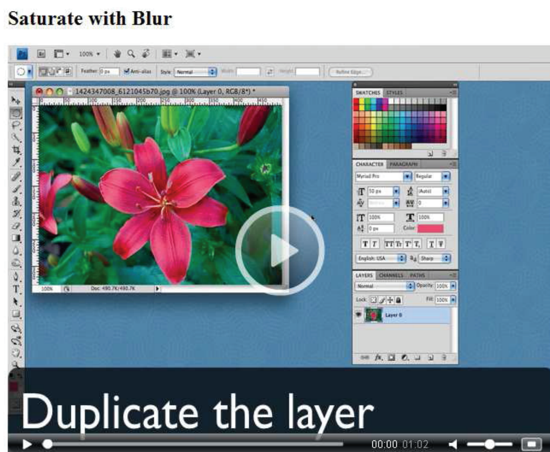


图 7-3　在使用了 Flowplayer 的 Internet Explorer 中显示视频

当然，要想顾全到既没安装 Flash，同时浏览器又不原生支持 video 的用户，我们还需要进一步的改进。我们将会在另一段中嵌入下载链接，让这类用户能够下载视频内容：

Download html5video/index.html

```
<section class="downloads">
  <header>
    <h3>Downloads</h3>
  </header>
  <ul>
    <li><a href="video/h264/01_blur.mp4">H264, playable on most platforms</a></li>
    <li><a href="video/theora/01_blur.ogv">OGG format</a></li>
    <li><a href="video/webm/01_blur.webm">WebM format</a></li>
  </ul>
</section>
```

我们可以借助 JavaScript 在浏览器支持 HTML5 video 的情况下隐藏这些下载链接，如下所示：

```
function canPlayVideo() {
  return !!document.createElement('video').canPlayType;
```

① Flash 的默认安全设置可能会禁止 Flowplayer 加载视频，除非播放器和视频都是由某个 Web 服务器提供的。

```
}
if(canPlayVideo()){
  $(#videos .downloads).hide();
}
```

JavaScript 的媒体内容 API

本章，我们泛泛地讲解了关于 audio 和 video 元素的 JavaScript API。完整的 API 可以检测浏览器可播放音频文件的类型，并且提供控制音频元素回放的方法。

在 7.3 节，我们建立了一个嵌有多个示例音频的页面。我们可以使用 JavaScript API 让这些音频同时（基本上）播放。下面是一种非常简单的处理方式：

Download **html5_audio/advanced_audio.html**

```
var element = $("<p><input type='button' value='Play all'/></p>")
element.click(function(){
  $("audio").each(function(){
    this.play();
  })
});

$("body").append(element);
```

代码中创建了一个“Play all”按钮，被按下的时候，这将影响到页面中全部的 audio 元素，并对每个 audio 元素都会调用 play()。

我们也可以对视频进行类似操作。对于视频，有播放、暂停甚至是查询当前播放进度的方法。

不过在写这本书的时候，JavaScript API 得到的支持度还不够。尽管如此，我们还是应该多了解规范*，寻找新的可能性。

* http://www.w3.org/TR/html5/video.html#media-elements。

这里用的检测技术同 7.3 节中的非常类似。这个例子中，我们允许用户将视频下载下来，以便过后在 iPod 或 iPad 上面观看，这样显得更有意义。

7.4.2 HTML5 视频的限制

目前 HTML5 视频面临着 3 种非常重要的形势，限制了其使用。

首先，HTML5 视频不支持流式传输和播放视频文件。用户已经习惯了跳转到一段视频中的

任意位置进行播放。在这点上，基于 Flash 的视频播放器是比较擅长的，这要归功于 Adobe 在将 Flash 作为一种视频传输平台方面做的巨大努力。要在 HTML5 video 中跳转播放，视频文件必须已完整加载至浏览器。这一点可能会随着时间的推移而发生改变。

其次，无法控制版权。像 Hulu①这类希望防止内容被盗版的网站就不能依赖于 HTML5 video。这种情况下，Flash 仍然是一种比较合适的解决方案。

最后，也是最重要的，对视频进行编码成本高耗时长。对于必须以多种格式对视频进行编码，这让 HTML5 video 的吸引力大大降低了。因此，我们看到很多网站提供了受版权保护的 H.264 格式的视频，以便在 iPod 和 iPad 上播放，同时混合使用了 HTML5 video 标签和 Flash。

这些弊端并不会拖 HTML5 的后腿，但却是我们在使用 HTML5 video 替代 Flash，来作为视频传输平台前需要提前了解的。

留意成人娱乐产业

从电子商务到 Flash 的蔓延，成人娱乐产业深深地影响了因特网技术。* HTML5 video 的到来让这个影响继续深入了下去。† 相比台式电脑和笔记本电脑，类似 iPhone 和 iPad 这类设备更私人化，而且不支持 Flash。因此，很多面向成人的站点已经开始将视频传输平台由 Flash 转向了 H.264 编码的 HTML5。有意思的是，虽然 HTML5 的视频当前没有任何版权保护机制，但他们似乎并不在意。

成人产业从来不惧怕风险，出于他们对技术的兴趣，我们很有可能看到在 HTML5 视频方面的一些喜人的发展。

* http://chicagopressrelease.com/news/in-tech-world-porn-quietly-leads-the-way。

† http://news.avn.com/articles/Joone-Points-to-HTML-5-as-Future-of-Web-Content-Delivery -401434.html。

7.4.3 音频、视频和可访问性

对于残障用户来说，上面那些回退方案都无法真正发挥作用。实际上，HTML5 规范明确指出了这一点。听觉有障碍的用户不会觉得能下载音频文件有多大意义，视觉有障碍的用户对于在浏览器之外还能够观看视频也不感兴趣。当我们将内容提供给用户的时候，应该尽可能提供可行

① 参见 http://www.hulu.com。

的替代解决方法。视频和音频文件应该有对应的字幕，以供用户阅读。对于自创的内容来说，如果在一开始就规划的话，字幕就很容易生成，因为可以直接取自你所编写的剧本。假如没办法生成一份字幕，可以考虑用一段总结性的语句来描述视频的重点内容。

Download html5video/index.html

```
<section class="transcript">
  <h2>Transcript</h2>
  <p>We'll drag the existing layer to the new button on the bottom of
    the Layers palette to create a new copy.</p>
  <p>Next we'll go to the Filter menu and choose "Gaussian Blur".
    We'll change the blur amount just enough so that we lose a little
    bit of the detail of the image.</p>
  <p>Now we'll double-click on the layer to edit the layer and
    change the blending mode to "Overlay". We can then adjust the
    amount of the effect by changing the opacity slider.</p>
  <p>Now we have a slightly enhanced image.</p>
</section>
```

我们可以选择隐藏字幕或者从视频主页链接它。总之，只要容易看到也容易跳转，那么就会起到很好的效果。

7.4.4 未来展望

浏览器中一流的音频支持为开发人员开辟了大量新的可能。JavaScript 开发的 Web 应用可以在无需借助 Flash 嵌入音频的情况下触发音效和提醒。原生音频支持可以让视频直接在 iPhone 等设备上播放，也给出了一种开放的、标准化的方式来进行 JavaScript 与视频的交互。最重要的，通过将视频和音频进行语义标注，让其易于识别，我们将可以像操作图片一样操作视频和音频。

第 8 章

柔化视觉体验

作为 Web 开发人员，我们总是希望用户界面能更加引人注目，而 CSS3 提供了很多种实现方法。我们可以在页面上使用自定义的字体，可以创建圆角元素和阴影，可以使用渐变的背景色，甚至可以旋转元素，使其看起来不那么单调枯燥。我们无需借助 Photoshop 或者其他图像软件，即可实现上述所有内容，而本章将介绍这些实现方法。首先，我们会为表单添加圆角效果，使其看起来更加柔和。然后，为即将迎来的贸易展览做一个横幅广告。通过此示例，我们将了解到如何添加阴影、旋转、渐变和透明度。最后，我们会讨论 CSS3 的`@font-face`属性的使用方法，该属性可以帮助我们在公司的博客上使用更加漂亮的字体。

本章，我们会详细介绍 CSS3 的如下属性。

- `border-radius[border-radius: 10px;]`——元素的圆角属性。[C4、F3、IE9、S3.2、O10.5]
- 对 RGBa 的支持 `[background-color: rgba(255,0,0,0.5);]`—— 使用 RGB 颜色和透明度，而非 16 进制颜色码。[C4、F3.5、IE9、S3.2、O10.1]
- 阴影效果 `[box-shadow: 10px 10px 5px #333;]`——为元素创建阴影。[C3、F3.5、IE9、S3.2、O10.5]
- 旋转 `[transform: rotate(7.5deg);]`——旋转任意元素。[C3、F3.5、IE9、S3.2、O10.5]
- 渐变 `[linear-gradient(top, #fff, #efefef);]`——创建渐变图片。[C4、F3.5、S4]
- `@font-face [@font-face { font-family: AwesomeFont; ]`

`src: url(http://example.com/awesomeco.ttf); font-weight: bold; }]`允许通过 CSS 使用特定字体。[C4、F3.5、IE5+、S3.2、O10.1]

8.1 实例 17：创建圆角

Web 页面上的元素默认都是长方形的。不论是表单域、表格，还是页面的区段，都有着块状的、棱角分明的外观。因此，多年来众多 Web 设计师一直致力于使用各种技术为这些元素添加

圆角效果，以便使用户界面看起来更柔和一些。

CSS3 可以轻松地实现圆角，Firefox 和 Safari 浏览器已经对此提供支持好长时间了。不过 Internet Explorer 还不支持这点，但我们可以轻松地解决这个问题。

8.1.1 圆角化登录表单

从目前项目的框架和原型图中我们看到，表单域都是圆角的。先用 CSS3 来实现这些圆角。我们的目标是做出如图 8-1 所示的页面。

图 8-1 圆角的表单

登录表单的 HTML 非常简单，如下所示：

Download css3roughedges/rounded_corners.html

```
<form action="/login" method="post">
  <fieldset id="login">
    <legend>Log in</legend>
    <ol>
      <li>
        <label for="email">Email</label>
        <input id="email" type="email" name="email">
      </li>
      <li>
        <label for="password">Password</label>
        <input id="password" type="password"
               name="password" value="" autocomplete="off"/>
      </li>
      <li><input type="submit" value="Log in"></li>
    </ol>
  </fieldset>
</form>
```

再来应用一些样式，使表单变得稍微好看一点：

Download css3roughedges/style.css

```
fieldset{
  width: 216px;
  border: none;
  background-color: #ddd;
}

fieldset legend{
  background-color: #ddd;
  padding: 0 64px 0 2px;
}

fieldset>ol{list-style: none;
  padding:0;
  margin: 2px;
}
fieldset>ol>li{
  margin: 0 0 9px 0;
  padding: 0;
}

/* 让输入框独立成行*/
fieldset input{
  display:block;
}
input{
  width: 200px;
  background-color: #fff;
  border: 1px solid #bbb;
}

input[type="submit"]{
  width: 202px;
  padding: 0;
  background-color: #bbb;
}
```

这些基本的样式去除了列表中的项目符号，并且保证输入框的大小都是一样的。在此基础上，我们开始对元素应用圆角样式。

8.1.2　特定于浏览器的选择器

由于 CSS3 规范还没生成终稿，各浏览器厂商分别加入了各自的独特功能，并在自己特定的功能前面加上了前缀。这样的话，即使有些功能还没纳入最终规范，浏览器厂商也可以通过这些前缀来引入它们。同时，由于这些功能并不严格符合规范，所以浏览器厂商可以在实现规范标准功能的同时保持自己的实现。大多数情况下，提供前缀的版本与 CSS 规范是吻合的，但是也有

不一样的时候。因此，麻烦丢给了开发人员，我们需要为每种浏览器分别定义不同的边界半径（border radius）。

Firefox 使用的选择器为：

Download css3roughedges/style.css

```
-moz-border-radius: 5px;
```

基于 Webkit 的浏览器，如 Safari 和 Chrome，使用的选择器为：

Download css3roughedges/style.css

```
-webkit-border-radius: 5px;
```

为了将页面上的输入控件都圆角化，需要定义如下所示的 CSS：

Download css3roughedges/style.css

```
input, fieldset, legend{
  border-radius: 5px;
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px;
}
```

把上述代码添加到 style.css 文件中，就会得到圆角效果。

8.1.3　回退

上面的内容可以在 Firefox、Safari、Google Chrome 浏览器中正常运行，但在 Internet Explorer 浏览器下无法正常显示。确实，我们的网站必须支持 Internet Explorer。因此，需要想个办法来解决在 IE 浏览器中无法正常显示的问题。

很早以前，Web 开发人员就使用圆角图片或其他手段来实现圆角了，不过我们希望实现起来越简单越好。可以使用 JavaScript 来检测圆角半径，然后使用任意圆角技术来实现圆角。比如，我们可以使用 jQuery、jQuery 的圆角插件以及专门用于文本输入框的修改版圆角插件。

8.1.4　检测对圆角的支持

我们的回退解决方案，与 4.3.2 节的回退方案极为类似。我们将引入 jQuery 库及插件，并检测浏览器是否支持 CSS3 属性。如果不支持，则需要激活插件。因此，我们需要检测 CSS 属性 `border-radius` 是否存在，并同时检查带有浏览器（如 `webkit` 和 `moz`）特定前缀的相应属性是否存在。

创建 corner.js，添加如下方法：

Download css3roughedges/corner.js

```
function hasBorderRadius(){
  var element = document.documentElement;
  var style = element.style;
  if (style){
    return typeof style.borderRadius == "string" ||
      typeof style.MozBorderRadius == "string" ||
      typeof style.WebkitBorderRadius == "string" ||
      typeof style.KhtmlBorderRadius == "string";
  }
  return null;
}
```

这样，我们就可以检测浏览器是否支持圆角了。接下来，添加对圆角的实现代码。非常幸运的是，我们有插件可以利用。

8.1.5 jQuery Corners

jQuery Corners[1]是一个小插件，通过使用 `div` 标签及样式对元素进行包装来实现圆角。但是无法实现表单域的圆角。不过稍微思考一下，即可使用 jQuery Corners 插件和 jQuery 来实现表单域的圆角。

> **这样做是否值得**
>
> 本例中，客户希望将所有浏览器都实现为圆角。不过，如果可能的话，我们应该始终将这些功能当做可选功能。有些人会认为柔化表单样式确实有很大好处，但我们首先应该考虑的是有多少用户的浏览器不支持基于 CSS 的圆角。如果访问者大多是 Safari 和 Firefox 的用户，那么我们就不值得花费时间编写和维护圆角功能检测代码和回退脚本。

首先，获取 jQuery Corners，并在 HTML 页面中链接它。同时，corner.js 也要链接进来：

Download css3roughedges/rounded_corners.html

```
<script src="jquery.corner.js" charset="utf-8" type='text/javascript'></script>
<script src="corner.js"  charset="utf-8" type='text/javascript'></script>
```

下面我们来添加实现圆角的代码。

---

① 参见 http://www.malsup.com/jquery/corner/。

8.1.6 自制表单圆角插件

这一节，我们将编写一个 jQuery 插件，使用该插件可以轻松地将所有表单域都圆角化。我们已在 3.3.3 节介绍了 jQuery 插件的编写方法，这里就不再描述该过程了。这里讨论的代码是部分基于 Tony Amoyal[1]的解决方案的一种实现，我们会将此插件的代码整体过一遍。

在 corner.js 中加入如下代码：

Download css3roughedges/corner.js

```
(function($){

  $.fn.formCorner = function(){
    return this.each(function() {
      var input = $(this);
      var input_background = input.css("background-color");
      var input_border = input.css("border-color");
      input.css("border", "none");
      var wrap_width = parseInt(input.css("width")) + 4;
      var wrapper = input.wrap("<div></div>").parent();
      var border = wrapper.wrap("<div></div>").parent();
      wrapper.css("background-color", input_background)
             .css("padding", "1px");
      border.css("background-color",input_border)
            .css("width", wrap_width + "px")
            .css('padding', '1px');
      wrapper.corner("round 5px");
      border.corner("round 5px");
    });
  };
})(jQuery);
```

我们所讨论的 jQuery 对象可能是一个元素，也可能是一组元素，我们将使用两个 div 包装它，实现圆角。首先，将内层 div 的颜色设置为与表单原始的背景色一致，去掉表单域的边框。然后，我们使用与表单边框原始颜色一致的背景色填充第二个 div，并留一些边距。这个边距实现了边框的轮廓。可以想象一下，有两张方形纸，一张是 4 英寸[2]宽的绿纸，一张是 3 英寸宽的红纸，当把小的放在大的正上面时，可以看到红纸的外面有一个绿色的边框。我们的实现与这个原理相同。

8.1.7 生成圆角

有了插件和对圆角的检测类库，我们就可以生成圆角了。

---

① 参见 http://www.tonyamoyal.com/2009/06/23/text-inputs-with-rounded-corners-using-jquery-without-image/。

② 1 英寸=0.0254 米。——编者注

在 corner.js 中加入如下代码：

Download css3roughedges/corner.js

```
Line 1 $(function(){
     2   if(!hasBorderRadius()){
     3     $("input").formCorner();
     4     $("fieldset").corner("round 5px");
     5     $("legend").corner("round top 5px cc:#fff");
     6   }
     7 });
```

我们将 3 个表单控件和控件组做成了圆角。在第 5 行，只将 `legend` 上方设置为了圆角，并将余下的切角设为白色。插件使用当前层的背景色作为切除部分的颜色，这在本例中并不适用。

如果浏览器支持 `border-radius` 属性，我们的插件就可以运行。如果不支持，之前添加的 CSS 就会生效。

8.1.8 微调

IE 对 `legend` 的处理稍有不同。我们可以通过微调样式来解决表单控件组的 `legend` 在 IE 浏览器中显示偏上的问题，使其在 IE 中的显示效果能与在 Firefox 和 Chrome 中一样。

Download css3roughedges/rounded_corners.html

```
<link rel="stylesheet" href="style.css" type="text/css" media="screen">
<!--[if IE]>
  <style>
    fieldset legend{margin-top: -10px }
  </style>
<![endif]-->
```

这样，在所有主流浏览器中看起来都较类似了。图 8-2 就是在 Internet Explorer 浏览器中的显示效果。

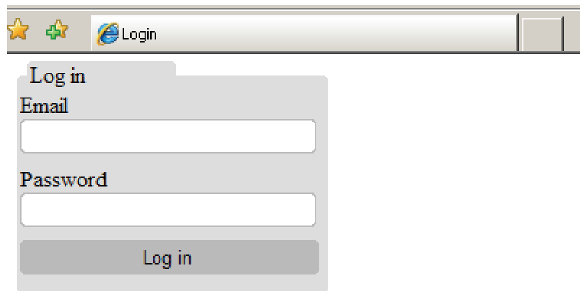


图 8-2 在 Internet Explorer 中的圆角表单

圆角增加了页面的柔和性，而且比较容易实现。需要注意的是，如同其他方面的设计一样，要适当地使用圆角，千万不要滥用。

8.2 实例 18：使用阴影、渐变和变换

尽管对圆角的关注占了很大篇幅，但这仅仅是我们探索 CSS3 的开始。我们可以为元素添加阴影，使它们相对于其他内容突显出来，可以使用渐变，使背景看起来更明确，也可以使用变换来旋转元素。让我们综合使用这些技术为即将到来的 AwesomeConf 做个横幅广告，AwesomeConf 是 AwesomeCo 每年都要举办的商业展览交流会。图形设计师提供了如图 8-3 所示的一个 PSD 图片。我们可以完全通过 CSS 实现标志、阴影，甚至调整透明度。人物的背景图片是唯一需要图片设计师提供给我们的东西。

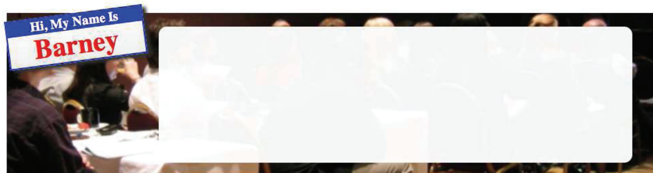


图 8-3　初始概念图，可以通过 CSS3 创建出来

8.2.1 基础结构

让我们先从 HTML 页面的基础结构做起：

Download css3banner/index.html

```
<div id="conference">
  <section id="badge">
    <h3>Hi, My Name Is</h3>
    <h2>Barney</h2>
  </section>

  <section id="info">
  </section>
</div>
```

对基础结构应用如下的样式：

Download css3banner/style.css

```
#conference{
  background-color: #000;
  width: 960px;
```

```
  float:left;
  background-image: url('images/awesomeconf.jpg');
  background-position: center;
  height: 240px;
}
#badge{
  text-align: center;
  width: 200px;
  border: 2px solid blue;
}

#info{
  margin: 20px;
  padding: 20px;
  width: 660px;
  height: 160px;
}


#badge, #info{
   float: left;
   background-color: #fff;
}


#badge h2{
  margin: 0;
  color: red;
  font-size: 40px;
}

#badge h3{
  margin: 0;
  background-color: blue;
  color: #fff;
}
```

对页面应用上述样式表后，可以使标志与内容区域并排显示，如图 8-4 所示。接下来让我们来修饰标志。



图 8-4 横幅广告初稿

8.2.2 增加渐变

修改标志的定义，将其背景色由白色改为从白到浅灰的渐变色。渐变样式在 Firefox、Safari 和 Chrome 中都会被支持，但是特定于 Firfox 的实现方法却不同。Chrome 和 Safari 都使用 Webkit 的语法，按最原始方案来实现，而 Firefox 使用的语法与 W3C 方案的类似。同样，需要使用浏览器前缀（参见 8.1.2 节）。①

Download css3banner/style.css

```
#badge{
  background-image: -moz-linear-gradient(
    top, #fff, #efefef
  );

  background-image: -webkit-gradient(
     linear,left top, left bottom,
     color-stop(0, #fff),
     color-stop(1, #efefef)
  );

  background-image: linear-gradient(
    top, #fff, #efefef
  );
}
```

Firefox 使用`-moz-linear-gradient`方法，该方法中需要指定渐变的起始点位置、起点的颜色和终点的颜色。

基于 Webkit 内核的浏览器允许我们设置中间节点的颜色。在本例中，我们只需要从白色渐变到灰色，但如果还需要渐变到其他颜色，那么只需要增加一个新的中间节点颜色就可以了。

8.2.3 给标志加上阴影

为标志加阴影也很简单。加上阴影可以使其看起来像是突显在横幅广告的前面，增加立体感。传统的做法是使用 Photoshop 给图片加上阴影或者以背景图片的形式插入阴影，而使用 CSS3 的`box-shadow`属性，可以轻松地为页面元素实现阴影效果。②

将该属性加入到样式表中，为标志增加阴影效果：

Download css3banner/style.css

```
#badge{
```

① 参见 http://dev.w3.org/csswg/css3-images/#linear-gradients。
② 参见 http://www.w3.org/TR/css3-background/#the-box-shadow。

```
  -moz-box-shadow: 5px 5px 5px #333;
  -webkit-box-shadow: 5px 5px 5px #333;
  -o-box-shadow: 5px 5px 5px #333;
  box-shadow: 5px 5px 5px #333;
}
```

box-shadow 属性有 4 个参数。第一个是水平偏移量，正数代表阴影会出现在目标元素的右边，负数代表阴影会出现在元素左边。第二个参数是垂直偏移量，正数代表阴影会出现在目标元素的下面，负数代表出现在目标元素的上面。

第三个参数是模糊半径。值为 0 的时候，阴影看起来非常清晰尖锐。值越大，阴影越模糊。最后一个参数定义了阴影的颜色。

我们需要不停地调整这些参数，来体会其工作原理，并找到那组适合我们的参数。当使用阴影时，需要首先调研一下现实生活中阴影的原理。用一束光照射在物体上，或者到外面去观察阳光下物体的阴影。这个调研非常重要，因为如果创建了一些不协调的阴影，尤其是对多个元素使用了不正确的阴影时，会使用户界面显得更加混乱。最简单的方法是，对每个元素的阴影都使用相同的设置。

文本上的阴影

除了对元素增加阴影样式，对文本也可以添加阴影效果。这与 box-shadow 类似：

```
h1{text-shadow: 2px 2px 2px #bbbbbb;}
```

需要定义 *X*、*Y* 方向的偏移量、模糊度和阴影的颜色。IE6、IE7 和 IE8 中也已经通过使用 Shadow 过滤器支持文本阴影：

```
filter: Shadow(Color=#bbbbbb,
  Direction=135,
  Strength=3);
```

这与在元素上应用阴影的方法是一样的。文本上的阴影会带来一种精致的效果，但是如果阴影太重的话会让文本变得不容易阅读。

8.2.4 旋转标志

我们可以使用 CSS3 的变换来旋转、缩放、倾斜元素，就像使用矢量图像程序 Flash、Illustrator、Inkscape 一样。[1]旋转可以使元素更突出，使 Web 页面看起来不再四四方方。让我们

① 参见 http://www.w3.org/TR/css3-2d-transforms/#transform-property。

稍微旋转一下标志，使之不再与条幅的直边平行。

Download css3banner/style.css

```
#badge{
  -moz-transform: rotate(-7.5deg);
  -o-transform: rotate(-7.5deg);
  -webkit-transform: rotate(-7.5deg);
  -ms-transform: rotate(-7.5deg);
  transform: rotate(-7.5deg);
}
```

使用 CSS3 进行旋转非常简单。我们只需要提供旋转的角度，页面就会自动渲染了。被旋转元素中的所有元素都会跟着一块旋转。

实现旋转与实现圆角一样简单，但是不要滥用它。界面设计的目标是使用户界面更加便于使用。如果旋转了包含很多内容的元素，一定要确保用户阅读内容的时候不用朝一个方向太过扭头。

8.2.5 调节背景的透明度

在文本后面加入半透明层来调节透明度的方法比较典型，很早前平面设计师就开始使用了。这种方法通常需要在 Photoshop 中创建一幅完整的图片或者使用 CSS 在另一个元素的上面放置一个透明的 PNG 层。CSS3 通过一种新的语法提供了对背景色透明度的支持。

初次接触 Web 开发时，大家都学习过使用十六进制颜色码定义颜色，用两个数字对来表示红色、绿色和蓝色。00 就是都关闭，即没有，FF 就是都开启。因此红色就是 FF0000，也可以理解为红色开启、蓝色和绿色关闭。

CSS3 引入了 rgb 和 rgba 函数。rgb 函数与十六进制表示法类似，但使用 0~255 的数字来表示每种颜色。可以定义红色为 rgb（255,0,0）。

rgba 函数与 rgb 一样，只不过多了一个参数来定义不透明度（0~1）。如果取值为 0，就什么颜色也看不到，因为是完全透明的。如果想得到半透明的白色区域，可以加入如下样式：

Download css3banner/style.css

```
#info{
  background-color: rgba(255,255,255,0.95);
}
```

有时用户显示器的对比度设置会影响上述透明度的效果，因此必须在多种显示设置情况下试验，才能保证得到一致的显示效果。

让我们将横幅广告的 info 区域圆角化：

Download css3banner/style.css

```
#info{
  moz-border-radius: 12px;
  webkit-border-radius: 12px;
  o-border-radius: 12px;
  border-radius: 12px;
}
```

应用上述样式后，条幅在 Safari、Firefox 和 Chrome 下都能显示得非常漂亮了。接下来让我们完成在 Internet Explorer 中的样式设置。

8.2.6 回退

本节中使用的技术，在 IE9 中是有效的，但是在 IE6、IE7 和 IE8 中都有一些问题！需要使用微软的 DirectX 过滤器来解决。这意味着我们需要依赖于条件声明加载一个只针对 IE 浏览器的样式表。我们还需要使用 JavaScript 创建一个 section 元素，然后使用 CSS 来渲染它，因为这些版本的 IE 不能原生识别该元素。

Download css3banner/index.html

```
    <!--[if lte IE 8]>

      <script>
      document.createElement("section");
      </script>

      <link rel="stylesheet" href="ie.css" type="text/css" media="screen">

    <![endif]-->
  </head>
  <body>
    <div id="conference">
      <section id="badge">
        <h3>Hi, My Name Is</h3>
        <h2>Barney</h2>
      </section>

      <section id="info">
      </section>
    </div>

  </body>
</html>
```

DirectX 过滤器在 IE6、IE7、和 IE8 中都能工作，但是在 IE8 中的触发方式不同。因此，需要对每个触发器定义两遍。下面我们来看下如何旋转元素。

8.2.7 旋转

我们可以使用这些过滤器来旋转元素，但是没有仅仅定义旋转角度那么简单。为了达到我们想要的效果，需要使用 Matrix 过滤器，并指定我们需要角度的正弦余弦。具体来讲，要给函数传递参数余弦、正弦的相反数、正弦和余弦（与第一个参数相同），如下所示：[①]

Download css3banner/filters.css

```
filter:  progid:DXImageTransform.Microsoft.Matrix(
  sizingMethod='auto expand',
    M11=0.9914448613738104,
    M12=0.13052619222005157,
    M21=-0.13052619222005157,
    M22=0.9914448613738104
  );

-ms-filter: "progid:DXImageTransform.Microsoft.Matrix(
    sizingMethod='auto expand',
      M11=0.9914448613738104,
      M12=0.13052619222005157,
      M21=-0.13052619222005157,
      M22=0.9914448613738104
    )";
```

太复杂了？是的，尤其是与前面的例子相比时。别忘了，我们前面希望的转动角度是−7.5 度。因此，本例中负角的正弦为负数，而我们需要一个正数。

数学是挺难懂的，让我们接着实现渐变吧。

8.2.8 渐变

IE 浏览器的渐变过滤器与其他浏览器中的类似，只是我们需要加入较多的代码。提供起点颜色和终点颜色，就可以实现渐变了：

Download css3banner/filters.css

```
filter: progid:DXImageTransform.Microsoft.gradient(
   startColorStr=#FFFFFF, endColorStr=#EFEFEF
);
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(
   startColorStr=#FFFFFF, endColorStr=#EFEFEF
)";
```

与其他浏览器不同，这里我们把渐变直接应用在元素上，而不是应用在 `background-image` 属性上。

---

① 使用 2×2 矩阵进行线性变换。

让我们继续使用这个过滤器定义 info 区域背景色的透明度。

8.2.9　透明度

渐变过滤器可以在起点和终点的颜色码上增加一个额外的十六进制值，使用开始的两位定义透明度。下面的代码实现的效果与我们想要的极为类似：

Download css3banner/filters.css

```
background: none;
filter:
   progid:DXImageTransform.Microsoft.gradient(
   startColorStr=#BBFFFFFF, endColorStr=#BBFFFFFF
);

-ms-filter: "progid:DXImageTransform.Microsoft.gradient(
  startColorStr='#BBFFFFFF', EndColorStr='#BBFFFFFF'
)";
```

这 8 位十六进制颜色码与 rgba 函数很类似，只是透明度值放在了最前面而不是最后面。因此，它们的顺序是透明度、红色、绿色和蓝色。

我们需要移走元素上的背景属性（background），使其兼容 IE7。如果大家已经跟着上述过程创建了样式表，就会发现上面的代码还无法正常显示，后面我们会解决这个问题。

8.2.10　整合

在 IE 浏览器里使用这些过滤器的一个更大的困难是，无法分开定义它们。要对一个元素定义多个过滤器，我们必须将多个过滤器定义为一个以逗号分隔的列表。下面就是 IE 浏览器中样式表的实际组织方式：

Download css3banner/ie.css

```
#info{
  background: none;
  filter:
     progid:DXImageTransform.Microsoft.gradient(
     startColorStr=#BBFFFFFF, endColorStr=#BBFFFFFF
  );
  -ms-filter: "progid:DXImageTransform.Microsoft.gradient(
    startColorStr='#BBFFFFFF', EndColorStr='#BBFFFFFF'
  )";
}
#badge{
  filter:
    progid:DXImageTransform.Microsoft.Matrix(
```

```
        sizingMethod='auto expand',
          M11=0.9914448613738104,
          M12=0.13052619222005157,
          M21=-0.13052619222005157,
          M22=0.9914448613738104
        ),
        progid:DXImageTransform.Microsoft.gradient(
           startColorStr=#FFFFFF, endColorStr=#EFEFEF
        ),
        progid:DXImageTransform.Microsoft.Shadow(
          color=#333333, Direction=135, Strength=3
        );

      -ms-filter: "progid:DXImageTransform.Microsoft.Matrix(
       sizingMethod='auto expand',
         M11=0.9914448613738104,
         M12=0.13052619222005157,
         M21=-0.13052619222005157,
         M22=0.9914448613738104
       ),
       progid:DXImageTransform.Microsoft.gradient(
          startColorStr=#FFFFFF, endColorStr=#EFEFEF
       ),
       progid:DXImageTransform.Microsoft.Shadow(
         color=#333333, Direction=135, Strength=3
       )";
  }
```

为了得到预期的效果，加了好多代码。但事实证明，这些属性是可用的。观察图 8-5，会发现已经很接近我们期望的效果了。剩下需要做的仅仅是将 `info` 区域圆角化，可以参考 8.1 节查看如何实现。

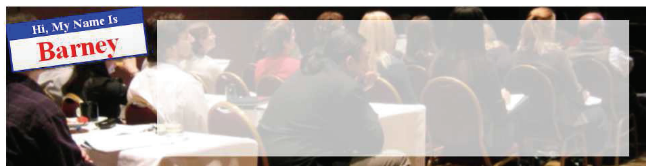


图 8-5 Internet Explorer8 中的横幅广告

尽管这些过滤器比较笨重，还有一点点古怪，但是为了给 IE 用户提供几乎一致的用户体验，在以后的项目中我们还会经常使用它们。

请注意，本节中谈论的一些效果都是介绍性的，当我们创建初始的样式表时，需要确保定义了背景色，以确保文字的可读性。这样的话，即使浏览器无法解析 CSS3 语法，仍可以显示页面并使其具有可读性。

8.3　实例 19：使用实用的字体

排版风格对用户体验来说非常重要。本书的字体也是精挑细选的，是由懂得如何选用合适字体及合适间距的专家认真选取后确定下来的，目的在于使读者获得轻松的阅览体验。这些对于网站来说同样重要。

在向读者传递信息时，我们选取的字体会影响到读者对信息的理解。图 8-6 所示的字体非常适合一支热情的重金属乐队。但是作为本书的封面字体就不太合适了，如图 8-7 所示。



图　8-6



图　8-7

由此可以看出，选择适合于文章内容的字体真的非常重要。Web 上字体的缺点是开发者可选择的余地太少，只有少量字体可用，俗称“websafe”字体。这些字体都是在大多数用户操作系统中被广泛使用的字体。

要解决这个问题，我们通常会将字体做成图片，然后直接加入到页面中或者通过 CSS 的方法将其当做背景图片或者 sIFR①，使用 Flash 渲染字体。CSS3 的字体模块提供了更好的实现方式。

8.3.1　@font-face

`@font-face` 指令实际上是在 CSS2 规范中引入的，并得到了 Internet Explorer 5 的支持。然而，微软的实现方式是使用一种叫做 Embedded OpenType（EOT）的字体格式，而现在大部分字体都是 TrueType 或者 OpenType 格式。当前的其他浏览器都支持 Open Type 和 True Typep 字体。

字体和版权

有些字体是收费的，像使用商业图片或者其他受版权保护的资料一样，需要获得版权和许可才可以在自己的网站上使用。如果购买了某套字体，需要在权利范围之内使用在自己网站的 logo 或者图片上。这叫做使用许可（usage right）。然而，`@font-face` 带来了一种不同的版权管理方式——分发权利。

① 参见 http://www.mikeindustries.com/blog/sifr。

当在页面上嵌入一种字体时，用户就需要下载该字体，意味着你的网站正在分发字体给别人。你需要确认正在使用的字体版权绝对允许这样做。

Typekit*有含很多字体的带有版权的字体库，并提供工具和代码让我们能轻松地将字体集成到自己的网站中。但是他们不免费提供服务，不过如果我们需要使用某种字体，其收费是非常便宜的。

Google 提供了 Googe Font API†，与 Typekit 类似，但是它仅提供开源字体。

这些服务都是通过 JavaScript 加载字体，因此需要开发者注意，要保证禁用 JavaScript 的用户页面的可读性。

只要记得一点，对待字体要如同对待其他资产一样，这样就不会遇到任何麻烦了。

* http://www.typekit.com/。
† http://code.google.com/apis/webfonts/。

AwesomeCo 的销售主管要求对公司的印刷品和网站使用统一的字体。网站需要引入一种免费的简单字体 Garogier。作为一次试验，我们会将这一字体应用于 2.1 节中创建的博客示例上。这样，大家就可以看到该字体的实例应用了。



小乔爱问……

如何转换自定义的字体?

如果开发人员开发了自定义的字体或者购买了一种字体，需要使其在多种格式下生效，那么可以借助 FontSquirre 网站上面的一个转换器（converter*），它可以提供转换后的字体以及我们需要的带有`@font-face`代码的样式表。但是需要确认我们的字体版权是否允许这么做。

* http://www.fontsquirrel.com/fontface/generator。

8.3.2 字体格式

字体拥有多种格式，目标浏览器会决定将什么格式的字体呈现给用户。

格式及浏览器支持情况

- Embedded OpenType（EOT）[IE5–8]

- TrueType（TTF）[IE9、F3.5、C4、S4]
- OpenType（OTF）[IE9、F3.5、C4、S4、O10.5]
- Scalable Vector Graphics（SVG）[IOS]
- Web Open Font（WOFF）[IE9、F3.6]

Internet Explorer 9 之前的浏览器只支持一种格式，即 Embedded OpenType（EOT）。其他浏览器则对 TrueType 和 OpenType 两种格式都能提供很好的支持。

Microsoft、Opera 和 Mozilla 联合起来创建了支持无损压缩的 Web Open Font 格式，同时对字体制作者来说，优化了许可管理。

为了兼容所有这些浏览器，我们需要让字体兼容多种格式。

8.3.3　改变字体

FontSquirrel[①]网站中有很多种字体，如 TrueType、WOFF、SVG 和 EOT 格式的字体，其使用效果很好。

字体的使用包含两个步骤，即定义字体和将其应用在元素上。在博客的样式表中，添加如下代码：

Download css3fonts/style.css

```
@font-face {
  font-family: 'GarogierRegular';
  src: url('fonts/Garogier_unhinted-webfont.eot');
  src: url('fonts/Garogier_unhinted-webfont.woff') format('woff'),
       url('fonts/Garogier_unhinted-webfont.ttf') format('truetype'),
       url('fonts/Garogier_unhinted-webfont.svg#webfontew0qE009') format('svg');
  font-weight: normal;
}
```

首先，我们定义了字体系列，赋予该系列一个名字，然后提供字体的链接。我们将指向 Embedded OpenType 的链接放在了最前面，这样 IE 可以立即解析它，之后依次提供了其他链接。用户的浏览器会依次尝试，直到找到一个可以使用的格式。

定义了字体系列后，就可以在样式表中使用了。修改之前的字体样式，如下所示：

① 参见 http://www.fontsquirrel.com/fonts/Garogier 或本书的下载代码。

Download css3fonts/style.css

```
body{
  font-family: "GarogierRegular";
}
```

通过这个小小的改变，页面文字将以新字体显示，如图 8-8 所示。



图 8-8 使用了新字体的博客

应用一种字体对当前主流浏览器来说相对比较容易，但是我们必须考虑不支持这些属性的其他浏览器。

8.3.4 回退

我们已经对多版本的 IE 及其他浏览器提供了回退方案，但还必须确保在浏览器不支持 `@font-face` 属性时，页面仍可读。

我们提供了 Garogier 字体的替代版本，但是应用该字体的时候，却并没有指定任何备用字体。这就是说，当浏览器不支持 Garogier 字体时，只能使用浏览器的默认字体显示，这是不够理想的。

字体栈是一个有优先级的字体列表。我们可以首先定义最希望用户看到的字体，然后定义其他字体作为适合的备用字体。

当定义字体栈的时候，需要额外花些时间来确认合适的备用字体。字符的间距、笔触的宽度和整体的外观都应该是相似的。UnitInteractive 网站上有一篇关于这个问题的很棒的文章。[①]

如下修改字体：

---

① 参见 http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks/。

Download css3fonts/style.css

```
font-family: "GarogierRegular", Georgia,
             "Palatino", "Palatino Linotype",
             "Times", "Times New Roman", serif;
```

这里提供了很多备用字体，可以帮助我们维持页面的相似外观。即使这样做了，也不能保证在任何情况下都能完美运行，但是至少比仅仅依赖默认字体要好，因为默认字体有时候是很不适合阅读的。

字体在使页面更有吸引力和更加易读方面会大有帮助，因此建议大家在实际工作中多多试验。现在已有很多免费的和商用的字体可供我们使用。

8.3.5 未来展望

本章，我们探讨了一些使用 CSS3 替换传统 Web 开发技术的方法，但是这些技术还只是皮毛。CSS3 规范还引入了 3D 变换甚至简单的动画，也就是说我们可以利用样式表向用户提供交互信息，而不是使用 JavaScript。就像我们使用:hover 时一样。

此外，一些浏览器已经支持复合背景图片和渐变边框。最后，也请关注一下对网页内容本身的改进，如滚动的头部和尾部以及对页码的支持。

在 CSS3 模块完成之后，我们便可以更加简单容易地为用户创建更丰富的、更完美的和更受欢迎的界面元素。因此，一定要多加关注新属性。

第 9 章

客户端数据的使用

前面讨论了 HTML5 和 CSS3 的标记，现在我们将注意力转向与 HTML5 相关的技术和功能。拿跨文档消息机制和离线支持两个功能来说，它们可以辅助我们实现跨域信息交互，为用户创建可离线使用的解决方案。

诸如 Web Storage、Web SQL Databases 和 Web Sockets 等一些标准都是从 HTML5 规范中剥离出来的，而像 Geolocation 等其他一些标准甚至从来都没有在规范中出现过，但浏览器厂商和开发人员却将其与 HTML5 联系到了一起，因为 HTML5 规范是同这些功能并肩推进的。

本书中的这一部分将涉及这些功能，并为现在就能使用的功能给予更高的关注。我们还将使用一章的篇幅来讨论未来趋势。让我们先从 Web Storage 和 Web SQL Storage 入手，这两个规范可以让我们在客户端存储数据。

还能想起来在什么情况下能对 cookie 竖起大拇指吗？我也想不起来。确实，自从 cookie 出现之后，对其处理一直都很令人头疼，但是我们容忍了这样的麻烦，因为毕竟它是过去唯一能在客户端机器中存储数据的方式。使用 cookie 的话，我们需要为其命名，还要设置其失效时间。

其中涉及一组以函数形式存在的 JavaScript 代码，用函数封装后我们无需考虑其工作原理，就像这样：

Download **html5_localstorage/setcookie.js**

```
//访问 http://www.javascripter.net/faq/settinga.htm
function SetCookie(cookieName,cookieValue,nDays) {
 var today = new Date();
 var expire = new Date();
 if (nDays==null || nDays==0) nDays=1;
 expire.setTime(today.getTime() + 3600000*24*nDays);
 document.cookie = cookieName+"="+escape(cookieValue)
                 + ";expires="+expire.toGMTString();
}
```

抛开难记的语法不说，安全因素也是需要考虑的。一些站点使用 cookie 存储用户的浏览行为，

所以有的用户会禁用 cookie。

HTML5 引入了在客户端存储数据的一些新选择：Web Storage[①]（使用 `localStorage` 或 `sessionStorage`）和 Web SQL Databases[②]。两者使用起来都很方便，难以置信得强大，具有合理的安全性。最重要的是，目前一些浏览器已经实现了这两个功能，包括 iOS 的 Mobile Safari 和 Android 2.0 的 Web 浏览器。然而，它们不属于 HTML5 规范，而是剥离出来形成了各自独立的规范。

在客户端和服务器数据共享方面，虽然 `localStorage`、`sessionStorage` 和 Web SQL Databases 不能替代 cookie，例如 Web 框架还需要使用 cookie 来维护请求间的状态信息，但是我们可以使用它们保存仅用户关心的内容，如外观设置或参数设置等。使用它们也可以很方便地创建运行在浏览器中的移动应用，而这种移动应用的运行无需连接到因特网。当前很多 Web 应用都是调用服务器来保存用户数据的，但使用这些新的存储机制的话，因特网连接就不再是绝对的依赖条件了。用户数据可以在本地存储并在需要时进行备份。

当我们将这些方法与 HTML5 的离线功能结合起来的时候，便可以在浏览器中建立可跨各种平台运行的完整的数据库应用，这些平台包括个人电脑、iPad 以及 Android 手机等。通过本章的介绍，我们可以学到如何使用这些技术来进行用户设置的持久化，以及如何建立一个简单的留言数据库。

本章涉及如下功能。

- `localStorage`——以键/值对的形式存储数据，与某个域绑定，数据可跨浏览器会话保持。[C5、F3.5、S4、IE8、O10.5、IOS、A]
- `sessionStorage`——以键/值对的形式存储数据，与某个域绑定，浏览器会话结束时被清除。[C5、F3.5、S4、IE8、O10.5、IOS, A]
- Web SQL Databases——完整的关系数据库，支持通过事务进行表创建、插入、更新、删除和选择操作。与某个域绑定，数据可跨浏览器会话保持。[C5、S3.2、O10.5、IOS3.2、A2]
- Offline Web Applications（离线 Web 应用）——定义缓存文件是为了方便离线使用，以便应用能够运行在没有因特网连接的环境下。[C4、S4、F3.5、O10.6、IOS3.2、A2]

9.1 实例 20：使用 `localStorage` 保存参数设置

对于希望在客户端机器上持久保持数据的开发人员开说，`localStorage` 机制提供了一种非常简便的方法。`localStorage` 机制简单来说就是嵌入在 Web 浏览器中的名/值对存储。

---

① 参见 http://www.whatwg.org/specs/web-apps/2007-10-26/#storage。
② 参见 http://www.whatwg.org/specs/web-apps/2007-10-26/#sql。

由 localStorage 存储的信息可跨浏览器会话保持，由于它被限制在当前访问的域中，所以不可被其他站点读取。[①]

AwesomeCo 正在开发一款新的客户服务门户产品，允许用户更改文本大小、背景以及站点的文字颜色。我们将用 localStorage 来实现该功能，以便当我们保存设置的时候，配置数据会从一个浏览器会话保持到另外一个。最后完成的时候，效果如图 9-1 所示。

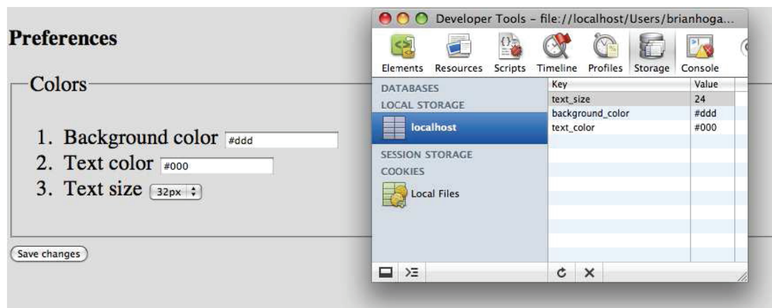


图 9-1　借助 localStorage，用户的参数值存储在了本地

9.1.1　创建参数表单

现在，我们使用 HTML5 语义标记和在第 3 章中所学的一些新表单控件来巧妙地制作一份表单。我们会允许用户更改前景色、背景色并调整字体大小。

Download html5_localstorage/index.html

```
<p><strong>Preferences</strong></p>
<form id="preferences" action="save_prefs"
      method="post" accept-charset="utf-8">
  <fieldset id="colors" class="">
    <legend>Colors</legend>
    <ol>
      <li>
        <label for="background_color">Background color</label>
        <input type="color" name="background_color"
               value="" id="background_color">
      </li>
      <li>
        <label for="text_color">Text color</label>
        <input type="color" name="text_color"
               value="" id="text_color">
      </li>
```

① 进行本地开发时要特别注意。举例来说，如何我们在 localhost 上进行开发，会很容易引起变量混淆。

```
      <li>
        <label for="text_size">Text size</label>
        <select name="text_size" id="text_size">
          <option value="16">16px</option>
          <option value="20">20px</option>
          <option value="24">24px</option>
          <option value="32">32px</option>
        </select>
    </ol>

  </fieldset>

  <input type="submit" value="Save changes">
</form>
```

我们将用 HTML 颜色代码来表示颜色。

9.1.2　保存和加载设置

通过 JavaScript 访问 `window.localStorage()`对象，即可使用 `localStorage`。设置属性名和属性值的方式非常简单，如下所示：

Download html5_localstorage/index.html

```
localStorage.setItem("background_color", $("#background_color").val());
```

获取属性值的方式也很简单：

Download html5_localstorage/index.html

```
var bgcolor = localStorage.getItem("background_color");
```

下面我们创建一个方法，用来保存表单中的所有设置：

Download html5_localstorage/index.html

```
function save_settings(){
  localStorage.setItem("background_color", $("#background_color").val());
  localStorage.setItem("text_color", $("#text_color").val());
  localStorage.setItem("text_size", $("#text_size").val());
  apply_preferences_to_page();
}
```

接下来创建一个类似的方法，从 `localStorage` 系统中加载数据，并填写到表单相应域中：

Download html5_localstorage/index.html

```
function load_settings(){
  var bgcolor = localStorage.getItem("background_color");
  var text_color = localStorage.getItem("text_color");
  var text_size = localStorage.getItem("text_size");
```

```
  $("#background_color").val(bgcolor);
  $("#text_color").val(text_color);
  $("#text_size").val(text_size);

  apply_preferences_to_page();
}
```

该方法还会调用另一个方法，用来将设置应用到页面上。下面我们来讨论这个方法。

9.1.3　应用设置

现在可以从 `localStorage` 中获取设置信息了，接下来需要将其应用到页面上。这里所涉及的参数设置全部与 CSS 相关，因此可以使用 jQuery 来修改任意元素的样式。

Download html5_localstorage/index.html

```
function apply_preferences_to_page(){
  $("body").css("backgroundColor", $("#background_color").val());
  $("body").css("color", $("#text_color").val());
  $("body").css("fontSize", $("#text_size").val() + "px");
}
```

最后，我们要设置在文档加载完成的时候，触发这一系列事件。

Download html5_localstorage/index.html

```
$(function(){

  load_settings();

  $('form#preferences').submit(function(event){
    event.preventDefault();
    save_settings();
  });
});
```

9.1.4　回退

只有最新版的 IE、Firefox、Chrome 和 Safari 才支持 `localStorage` 方法，因此我们需要为旧版本浏览器提供回退方案。我们有多种可选方式，可以将信息保存至服务器，也可以在客户端使用 cookie 来保存参数设置。

1. 服务器端存储

如果系统有用户账户的功能，可以考虑将参数设置数据当做用户记录保存在应用中。当用户登录的时候，首先检测是否存在客户端配置信息，如果没有，就从服务器获取。通过这种方式，用户的设置可以跨浏览器、跨计算机保存。

要采用服务器保存数据的方式，请确保表单数据能够正常提交至服务器（如果已经不支持 cookie 了，就请不要再禁用 JavaScript 默认的提交行为了）。

如果用户禁用了 JavaScript，那么有效的数据保持方法就只剩下服务器端存储这一种了。因为我们可以在编写应用时指定让其从数据库中获取设置信息，而不是从 `localStorage` 的散列表中获取。而且对于存储数据大于 4 KB 的情况来说，也只能使用这一种方式，因为 cookie 的数据存储量最大限制为 4 KB。

2. cookie和JavaScript

cookie 和 JavaScript 的紧密结合，可以为我们打造一种非常不错的回退方案。Quirksmode[1]是一款著名的 cookie 脚本，我们可以通过 Quirksmode 建立自己的 `localStorage` 回退方案。

检测浏览器是否支持 `localStorage` 非常简单。只需要检测 `window` 对象上是否存在 `localStorage` 方法即可：

Download **html5_localstorage/index.html**

```
if (!window.localStorage){
}
```

接下来，需要编写写入 cookie 的方法。在这里我们借用 Quirksmode 的内容。将这些 JavaScript 函数以大括号括起来添加到我们自己的脚本中：

Download **html5_localstorage/index.html**

```
function createCookie(name,value,days) {
  if (days) {
    var date = new Date();
    date.setTime(date.getTime()+(days*24*60*60*1000));
    var expires = "; expires="+date.toGMTString();
  }
  else var expires = "";
  document.cookie = name+"="+value+expires+"; path=/";
}

function readCookie(name) {
  var result = ""
  var nameEQ = name + "=";
  var ca = document.cookie.split(';');
  for(var i=0;i < ca.length;i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1,c.length);
    if (c.indexOf(nameEQ) == 0){
```

① 参见 http://www.quirksmode.org/js/cookies.html。

```
        result = c.substring(nameEQ.length,c.length);
      }else{
        result = "";
      }
    }
    return(result);
  }
```

最后，我们希望创建 `localStorage` 对象，让它在后台使用 cookie。下面的这个示例虽然不成熟，不过足以将流程演示完整：

Download html5_localstorage/index.html

```
Line 1  localStorage = (function () {
   -      return {
   -        setItem: function (key, value) {
   -          createCookie(key, value, 3000)
   5        },
   -
   -        getItem: function (key) {
   -          return(readCookie(key));
   -        }
  10      };
   -    })();
```

注意第 4 行，这里创建的 cookie 其有效期是从当前时间往后算起的 3000 天。由于无法创建永远不失效的 cookie，所以这里使用一个极长的时间来代表永远。

在 `localStorage` 对象之外，使用方式与原先无异。如果我们需要删除元素或者清空对象，则需要多一点创造性。乐观估计，不久的将来我们便可以摒弃这种劣质的兼容方案，完全依赖于浏览器的 `localStorage` 方法。

sessionStorage

要想让一些数据在用户关闭 Web 浏览器之后仍然能够保存，可以使用 `localStorage`。但有的时候，我们只需要在浏览器打开的时候保存数据，一旦浏览器关闭，就不需要了。这时就要用到 `sessionStorage`。`sessionStorage` 的用法与 `localStorage` 一样，只是 `sessionStorage` 中存储的信息会在浏览器会话结束的时候失效。编写代码的时候仿照获取 `localStorage` 对象的方法，获取 `sessionStorage` 对象。

```
sessionStorage.setItem('name', 'Brian Hogan');
var name = sessionStorage.getItem('name');
```

为 `sessionStorage` 创建回退方案也很简单，只需要确保我们创建的 cookie 能够在浏览器关闭时失效即可。

9.2 实例 21：在客户端关系数据库中保存数据

localStorage 和 sessionStorage 为我们提供了在客户端计算机上保存简单名/值对数据的一种简便方法，但有时候这是远远不够的。起初，在关系数据库中存储数据的能力是在 HTML5 规范中引入的。后来这部分发展成了一份独立的规范，名为 Web SQL Storage[①]。只要开发人员有编写 SQL 代码的基础，它使用起来就会非常顺手。为方便演示，这里将使用 Web SQL Storage 来创建、获取、更新和删除客户端数据库中的信息。

9.2.1 浏览器中的CRUD

所谓 CRUD，是"Create、Retrieve、Update 和 Delete"[②]的简称，即创建、获取、更新和删除，恰恰讲述了我们使用客户端数据库会做的事情。通过其规范和实现，我们可以对数据进行增删改查的操作。

AwesomeCo 希望用一款可以在销售人员外出时收集留言的简单应用来武装自己的销售队伍。这个应用建好以后，不仅可以让用户创建新留言，还可以使之更新和删除已有的留言。要更改已有的留言，我们就需要让用户能先从数据库中将其取出来。

下面的 SQL 语句就是我们要用到的。

类　型	语　句
创建留言	`INSERT INTO notes (title, note) VALUES("Test", "This is a note");`
获取所有留言	`SELECT id, title, note FROM notes;`
获取特定留言	`SELECT id, title, note FROM notes where id = 1;`
更新留言	`UPDATE notes set title = "bar", note = "Changed" where id =1;`
删除留言	`DELETE FROM notes where id = 1;`



小乔爱问……

Web SQL Databases 规范不是已经夭折了吗?

在 2010 年 11 月，维护这一规范的工作组宣称不再推进 Web SQL Databases，将注意力转向了 IndexedDB 规范。我们之所以还在这里讨论 Web SQL Databases，是因为基于 Webkit 的浏览器已经实现了此功能，包括所有 iOS 和 Android 设备、Safari 以及 Google Chrome。截至本书编写之时，还没有任何一种浏览器支持 IndexedDB，但我们的项目中马上可以使用到 Web SQL Databases，这或许是满足我们需求的正确之选。

---

① 参见 http://dev.w3.org/html5/webdatabase/。

② 也可以表示"Create、Read、Update 和 Destroy"，即创建、读取、更新和销毁。

9.2.2　留言的前端展现

留言系统的前端左侧是一个侧边栏，其中是罗列出留言的列表，右侧包含一个标题域和用于编写留言的大文本框。效果见图 9-2。

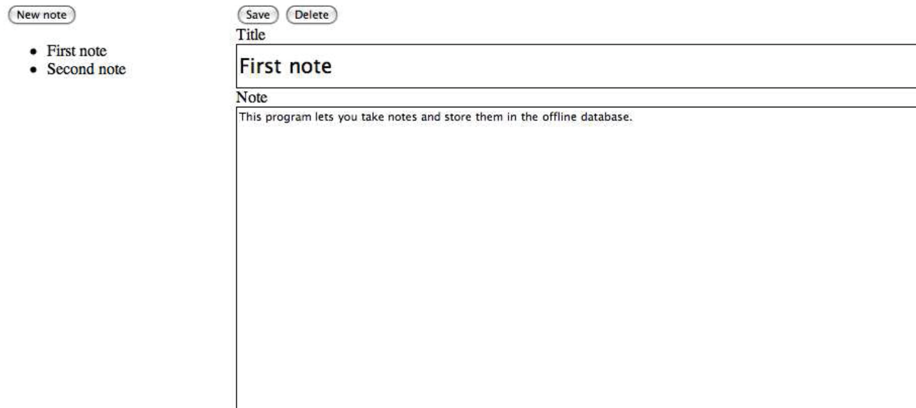


图 9-2　留言系统界面

首先，我们需要编写界面：

Download html5sql/index.html

```
<!doctype html>

<html>
  <head>
    <title>AwesomeNotes</title>
    <link rel="stylesheet" href="style.css">

  <script type="text/javascript"
    charset="utf-8"
    src=
    "http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
  </script>

    <script type="text/javascript"
      charset="utf-8" src="javascripts/notes.js">
    </script>

  </head>
  <body>
    <section id="sidebar">
      <input type="button" id="new_button" value="New note">
      <ul id="notes">
```

```
        </ul>
      </section>

      <section id="main">
        <form>
          <ol>
            <li>
              <input type="submit" id="save_button" value="Save">
              <input type="submit" id="delete_button" value="Delete">
            </li>
            <li>
              <label for="title">Title</label>
              <input type="text" id="title">
            </li>
            <li>
              <label for="note">Note</label>
              <textarea id="note"></textarea>
            </li>
          </ol>
        </form>
      </section>

    </body>
  </html>
```

我们使用 section 标签定义侧边栏和主体区域，并且为每个重要的用户界面控件指定了 ID，例如 Save（保存）按钮等。这样做可方便我们定位元素，以便为其附加事件监听器。

我们还需要一个样式表，这样才能达到示例图中的显示效果。style.css 如下所示：

Download **html5sql/style.css**

```
#sidebar, #main{
  display: block;
  float: left;
}

#sidebar{
  width: 25%;
}

#main{
  width: 75%;
}

form ol{
  list-style: none;
  margin: 0;
  padding: 0;
}
```

```
form li{
  padding: 0;
  margin: 0;
}

form li label{
  display:block;
}

#title, #note{
  width: 100%;
  font-size: 20px;
  border: 1px solid #000;
}

#title{
  height: 20px;
}

#note{
  height: 40px;
}
```

这份样式表禁用了项目符号，定义了文本域的大小，将元素分成了两列。至此，前端代码就完成了。接下来要做的是搭配 JavaScript 代码来让其发挥作用。

9.2.3 连接数据库

我们需要建立连接并创建一个数据库：

Download **html5sql/javascripts/notes.js**

```
// 数据库引用
var db = null;

// 创建到本地数据库的连接

connectToDB = function()
{
   db = window.openDatabase('awesome_notes', '1.0',
                                        'AwesomeNotes Database', 1024*1024*3);
};
```

首先，我们在脚本顶部声明了 db 变量。这样在我们创建的其余所有方法中它都会有效。[①]然后我们使用 `window.openDatabase` 方法声明了连接到数据库的方法。这里使用的参数有数据库名称、版本号、描述以及描述大小的参数。

---

① 这样做创建的是全局变量，但是通常不推荐使用。这里只是为了让 JavaScript 代码能够简洁易辨。

9.2.4 创建留言表

这里的留言表需要 3 列，如下所示。

字 段	描 述
id	留言的唯一标识。主键、整数、自增
title	留言的标题，以便于引用
Note	留言

下面编写一个方法来创建数据库表：

Download html5sql/javascripts/notes.js

```
createNotesTable = function()
{
  db.transaction(function(tx){
    tx.executeSql(
      "CREATE TABLE notes (id INTEGER \
       PRIMARY KEY, title TEXT, note TEXT)", [],
      function(){ alert('Notes database created successfully!'); },
      function(tx, error){ alert(error.message); } );
  });
};
```

我们在事务中触发 SQL 语句。该事务包含两个回调方法，一个针对执行成功的情况，一个针对执行失败的情况。这是以后类似动作中使用的通用模式。

注意，executeSql()方法的第二个参数是个数组。该数组用来将 SQL 中的占位符绑定到变量。这样便避免了拼接字符串，同其他语言中的 prepare 语句类似。此处，数组是空的，因为我们的示例中无需任何占位符。

现在第一个数据库表就建好了，下面我们要让这个应用真正做点实际的操作。

9.2.5 加载留言

我们希望应用加载的时候连接到数据库（如果数据库表不存在就创建一个）然后从数据库中获取所有的留言。

Download html5sql/javascripts/notes.js

```
// 从数据库的 notes 表中获取所有记录
fetchNotes = function(){
  db.transaction(function(tx) {
      tx.executeSql('SELECT id, title, note FROM notes', [],
        function(SQLTransaction, data){
          for (var i = 0; i < data.rows.length; ++i) {
```

```
                var row = data.rows.item(i);
                var id = row['id'];
                var title = row['title'];

                addToNotesList(id, title);
            }
        });
    });
};
```

该方法从数据库中获取所有记录。如果获取成功，就遍历所有记录，调用 addNoteToList 方法，该方法如下所示：

Download html5sql/javascripts/notes.js

```
//指定 id 和标题，将列表项添加到留言列表
addToNotesList = function(id, title){
  var notes = $("#notes");
  var item = $("<li>");
  item.attr("data-id", id);
  item.html(title);
  notes.append(item);
};
```

我们将记录的 ID 嵌入到自定义的数据属性中。然后在用户单击列表项的时候，我们会使用该 ID 定位数据库中的记录。然后将新创建的列表项连同 notes 的 ID 一起插入到前端的无序列表中。现在我们需要编写加载数据的代码，以便在用户在列表中进行选择的时候将其加载到表单中。

9.2.6 获取指定记录

我们可以为每个列表项都添加一个 click 事件，但实践经验告诉我们，有更好的方法，那就是监听整张无序列表中的单击事件，然后判断被单击的项。这样的话，如果列表中的项增加了（比如添加了一条留言），也无需为其再添加单击事件了。

在 jQuery 函数中，添加如下代码：

Download html5sql/javascripts/notes.js

```
$("#notes").click(function(event){
  if ($(event.target).is('li')) {
    var element = $(event.target);
    loadNote(element.attr("data-id"));
  }

});
```

它会调用 loadNote()方法，该方法如下：

Download html5sql/javascripts/notes.js

```
loadNote = function(id){
  db.transaction(function(tx) {
    tx.executeSql('SELECT id, title, note FROM notes where id = ?', [id],
      function(SQLTransaction, data){
        var row = data.rows.item(0);
        var title = $("#title");
        var note = $("#note");

        title.val(row["title"]);
        title.attr("data-id", row["id"]);
        note.val(row["note"]);
        $("#delete_button").show();

      });
  });
}
```

此函数同先前的 `fetchNotes()`很像。其中触发一条 SQL 语句，然后处理成功的情况。这里的语句中包含一个问号占位符，其实际值以数组元素的形式写在第二个参数中。

当查找到某条记录的时候，就将其显示在表单中。该方法同时还会启用 Delete 按钮，并将记录的 ID 嵌入到自定义数据属性中，这样便于处理更新操作。Save 按钮会检测 ID 是否存在。如果存在，就更新记录。如果不存在，就认为它是一条新记录。下面我们来编写这段逻辑。

9.2.7 插入、更新和删除记录

当用户单击 Save 按钮的时候，要触发代码来进行新数据插入或者更新已有数据。我们将在 Save 按钮上添加一个单击事件处理程序，方法是将下面的代码添加到 jQuery 函数中：

Download html5sql/javascripts/notes.js

```
$("#save_button").click(function(event){
  event.preventDefault();
  var title = $("#title");
  var note = $("#note");

  if(title.attr("data-id")){
    updateNote(title, note);
  }else{
    insertNote(title, note);
  }
});
```

此方法检测表单中标题域的 `data-id` 属性。如果没有 ID，表单认为用户在添加新记录，便会触发 `insertNote` 方法，此方法如下：

Download html5sql/javascripts/notes.js

```
insertNote = function(title, note)
{
   db.transaction(function(tx){
      tx.executeSql("INSERT INTO notes (title, note) VALUES (?, ?)",
                    [title.val(), note.val()],
        function(tx, result){
         var id = result.insertId ;
         alert('Record ' + id+ ' saved!');
         title.attr("data-id", result.insertId );
         addToNotesList(id, title.val());
         $("#delete_button").show();

        },
        function(){
          alert('The note could not be saved.');
        }
      );
   });
};
```

insertNote()方法将记录插入到数据库中，使用结果集的 insertid 属性获取刚插入记录的 ID。然后将其作为一种自定义数据属性应用到表单的 title 域，触发 addToNotesList()方法将留言添加到页面左侧的列表中。

接下来，我们需要处理更新事件。updateNote()方法同其他方法类似：

Download html5sql/javascripts/notes.js

```
updateNote = function(title, note)
{
   var id = title.attr("data-id");
   db.transaction(function(tx){
    tx.executeSql("UPDATE notes set title = ?, note = ? where id = ?",
                 [title.val(), note.val(), id],
      function(tx, result){
        alert('Record ' + id + ' updated!');
        $("#notes>li[data-id=" + id + "]").html(title.val());
      },
      function(){
        alert('The note was not updated!');
      }
    );
  });
};
```

update 语句成功执行后，我们通过 data-id 域中的刚更新过的 ID 值更新列表中的留言标题。

删除记录几乎同更新是一样的。我们需要如下所示的删除事件处理程序：

Download html5sql/javascripts/notes.js

```
$("#delete_button").click(function(event){
  event.preventDefault();
  var title = $("#title");
  deleteNote(title);
});
```

接着，编写删除方法。此方法不仅要从数据库中删除记录，而且还要在前端的侧边栏列表中也一并删除它：

Download html5sql/javascripts/notes.js

```
deleteNote = function(title)
{
   var id = title.attr("data-id");
   db.transaction(function(tx){
      tx.executeSql("DELETE from notes where id = ?", [id],
        function(tx, result){
         alert('Record ' + id + ' deleted!');
         $("#notes>li[data-id=" + id + "]").remove();
        },
        function(){
         alert('The note was not deleted!');
        }
      );
   });
};
```

现在剩下的工作只是清空表单了。这样做是为了避免在创建新的记录时意外地影响到现有的记录。

9.2.8 包装

我们的留言系统基本完成了，接下来只需要激活 New 按钮即可。按下该按钮会清空表单，这样用户便可以在编辑完一条留言后，创建新的留言。与先前的模式一样，先从 jQuery 函数中 New 按钮的事件处理程序开始：

Download html5sql/javascripts/notes.js

```
  $("#new_button").click(function(event){
    event.preventDefault();
    newNote();
  });
  //end:newbutton


  newNote();

});
```

接着，清空“title”域的 data-id 属性，从表单中移除对应值。在此，我们还会隐藏页面上的 Delete 按钮：

Download html5sql/javascripts/notes.js

```
newNote = function(){
  $("#delete_button").hide();
  var title = $("#title");
  title.removeAttr("data-id");
  title.val("");
  var note = $("#note");
  note.val("");
}
```

我们应该在 jQuery 函数内部调用 newForm 方法，这样当页面加载的时候表单就可用了。这样的话，Delete 按钮也就隐藏了。

所有的操作都完成了。我们的应用在 iPhone、Android 设备和安装有 Chrome、Safari 和 Opera 的计算机中运行。不过在 Firefox 中有一定几率可成功，而在 Internet Explorer 中却得不到支持。

9.2.9　回退

同其他的解决方案不同，对于 SQL 存储来说，还没有比较好的库可用。因此无法为 Internet Exlporer 用户提供原生支持。但是，如果你认为这种应用是有用的，那么可以说服用户使用 Google Chrome 来访问这个特殊的应用，因为 Google Chrome 在各种平台上都能使用。实际项目中这种方式并不少见，特别是对于需要同时在手机上访问的内部系统来说，要求更换浏览器是很正常的。

另一种可选方案是使用 Google Chrome Frame 插件[①]。在 HTML 页面中的 head 标签下添加：

Download html5sql/index.html

```
<meta http-equiv="X-UA-Compatible" content="chrome=1">
```

这个代码片段会被 Google Chrome Frame 插件读取，并为页面启用此功能。

如果要检测插件的存在性，然后在它不存在的情况下提醒用户安装，那么可以将下面的代码插入到 body 结束标签的上方：

Download html5sql/index.html

```
<script type="text/javascript"
 src=
 "http://ajax.googleapis.com/ajax/libs/chrome-frame/1/CFInstall.min.js">
</script>
```

① 参见 http://code.google.com/chrome/chromeframe/。

```
<script>

 window.attachEvent("onload", function() {
   CFInstall.check({
     mode: "inline", //默认
     node: "prompt"
   });
 });
</script>
```

这样可给用户一个安装插件的选择，以便他们能够正常使用我们的网站。

虽然对于面向广泛大众的应用来说，Google Chrome Frame 可能不是一种可行的解决方案，但是对于像我们刚刚建立的这种内部系统来说，效果还是不错的。可能会有一些相应的 IT 公司政策禁止这么做，不过这个问题就交给读者自己了，看在类似的情况下你会采取什么方法使这种方案获得批准？安装插件耗费的代价肯定要比自己编写 SQL 数据库系统要小很多。

9.3 实例 22：离线运行

有了 HTML5 离线功能[①]的支持，我们可以使用 HTML 和相关技术来建立在无因特网连接的情况下仍能正常使用的应用。特别是对于容易掉线的移动设备来说，开发这种功能的应用非常有用。

该技术支持 Firefox、Chrome 和 Safari，也支持 iOS 和 Android 2.0 设备，但对于 Internet Explorer 来说还没有可用的回退方案。

AwesomeCo 刚刚为其销售团队购买了一些 iPad，希望我们在 9.3 节中所开发的留言系统能够离线运作。多亏了 HTML 的 manifest 文件，这项工作变得轻而易举。

9.3.1 使用manifest定义缓存

为了实现离线运行，所有需要存储在客户端浏览器缓存中的 Web 应用的客户端文件都会以列表的形式存储在 manifest 文件中。应用涉及的所有文件都必须添加到这个列表中，以免运行出错。只有一种特殊情况，就是包含 manifest 的文件无需添加，它会被自动缓存。

建立一个名为 notes.manifest 的文件。其内容如下：

Download html5offline/notes.manifest

```
CACHE MANIFEST
# v = 1.0.0
/style.css
```

① 参见 http://www.w3.org/TR/html5/offline.html。

```
/javascripts/notes.js
/javascripts/jquery.min.js
```

我们可以修改其中的版本注释，以便浏览器获取最新版文件。如果对代码做了改动，我们需要相应地修改 manifest 文件。

另外，我们使用了在 Google 服务器上的 jQuery，因此若应用离线运行的话，就无法访问它了。因此我们需要下载 jQuery，并且修改 script 标签，让其从 javascripts 文件夹中加载 jQuery。

Download html5offline/index.html

```
<script type="text/javascript"
  charset="utf-8"
  src="javascripts/jquery.min.js">
</script>
```

接下来，需要将 manifest 文件链接到 HTML 文档中。实现方法是按如下方式修改 html 元素：

Download html5offline/index.html

```
<html manifest="notes.manifest">
```

至此全部操作就完成了。在此仅有一点小小的不足之处，因为必须通过 text/cache-manifest 的 MIME 类型使用，所以需要 Web 服务器来提供 manifest 文件。如果使用的是 Apache，可以在.htaccess 文件中这样设置 MIME 类型：

Download html5offline/.htaccess

```
AddType text/cache-manifest .manifest
```

只要向留言系统发送过一次请求，manifest 列表中的文件就会被下载并缓存起来。然后我们就可以切断网络连接，以离线方式随意访问系统了。

一定要多看规范。manifest 文件还有更多复杂的可用选项。例如，可以指定特定文件不能被缓存且永远不能被离线访问，这样做有利于忽略特定的动态文件。

9.3.2　manifest和缓存

以开发模式使用系统的时候，很可能需要在 Web 服务器上禁用一些缓存。很多 Web 服务器缓存文件的默认方式是，通过设置 header 告诉浏览器在指定的时间内不要抓取新文件副本。这样有助于在向 manifest 文件中添加信息的时候跟踪 bug。

如果使用的是 Apache，可以在.htaccess 文件中添加如下信息来禁用缓存：

Download html5offline/.htaccess

```
ExpiresActive On
ExpiresDefault "access"
```

上述代码会禁用整个目录的缓存，在实际发行产品的时候应该是不会这样做的。不过这样可以确保浏览器总是获取最新版本的 manifest 文件。

如果 manifest 中所列的某个文件有了改动，我们也需要修改 manifest 文件，方法是更改添加到注释中的版本号。

9.3.3 未来展望

`localStorage` 和 Web SQL Databases 之类的功能，可以让开发者开发出无需连接到 Web 服务器即可在浏览器中访问的系统。运行在 iPad 和 Android 设备上的应用也一样。当我们将这类功能和 HTML5 的 manifest 文件结合起来的时候，无需依赖特定平台，便可使用熟悉的工具建立离线富应用了。随着越来越多的浏览器开始对这些功能提供支持，开发人员对它们的利用率也会越来越高，可以创建可运行在多平台和设备上，可在本地存储数据，可在联网后进行数据同步的应用。

Web SQL Storage 的未来尚未可知。Mozilla 不准备在 Firefox 中实现 Web SQL Storage，同时 W3C 也换了一种选择转而推进 IndexedDB 规范的实现。在 11.5 节我们会对 IndexedDB 规范进行深入讨论。然而在 iOS 和 Android 设备上，Web SQL Storage 已经应用有段时间了，而且看上去还会持续下去。所以对于开发这类应用的人员来说，Web SQL Storage 规范会在其开发过程中起到很重要的作用。

第 10 章

使用其他 API 锦上添花

许多有趣的 API 都诞生于 HTML5 规范，但最终却发展成了独立的项目，而另外一些则与 HTML5 紧密联系，其中差异对于开发人员（甚至作者）来说，有时候真的很难言表。本章讨论的就是这类 API。我们会花费一点时间来研究旧的 HTML5 API，然后使用跨文档消息机制实现不同服务器间页面的通信。接着还会讨论两个非常强大的 API，即 Web Sockets 和 Geolocation，通过它们可以实现更具交互性的应用。

建立这些应用，会用到下列 API。

- History——管理浏览器历史记录。[C5、S4、IE8、F3、O10.1、IOS3.2、A2]
- 跨文档消息机制（Cross-document Messaging）——在窗口间传递加载自不同域的内容。[C5、S5、F4、IOS4.1、A2]
- Web Sockets——在浏览器和服务器间建立有状态的连接。[C5、S5、F4、IOS4.2]
- Geolocation——从客户端浏览器中获取经纬度。[C5、S5、F3.5、O10.6、IOS3.2、A2]

10.1 实例 23：维护历史记录

HTML5 规范引入了一个用于管理浏览器历史记录的 API[①]。在 5.2 节，我们为 AwesomeCo 的新主页创建了原型，在单击导航 tab 的时候，它会变换主页面内容。这种处理方式的唯一不足之处就是不支持浏览器的后退按钮。虽然可以借助一些辅助手段来处理，但如果使用 History API 的话问题最终可以得到较好地解决。

这样检测对此 API 的支持情况：

Download html5history/javascripts/application.js

```
function supportsHistory(){
  return !!(window.history && window.history.pushState);
}
```

每次需要使用 History 对象的时候，我们都要进行一次上述检测。

① 参见 http://www.w3.org/TR/html5/history.html。

10.1.1 保存当前状态

当用户浏览到新的 Web 页面时，浏览器会将其添加到历史记录中，而当用户打开新 tab 的时候，我们需要手动将其添加到历史记录中，如下所示：

Download html5history/javascripts/application.js

```
Line 1  $("nav ul").click(function(event){
-         target = $(event.target);
-         if(target.is("a")){
-           event.preventDefault();
5           if ( $(target.attr("href")).hasClass("hidden") ){
-
▶             if(supportsHistory()){
▶               var tab = $(target).attr("href");
▶               var stateObject = {tab: tab};
10▶             window.history.pushState(stateObject, tab);
▶             };
-
-             $(".visible").removeClass("visible").addClass("hidden").hide();
-             $(target.attr("href")).removeClass("hidden").addClass("visible").show();
15          };
-         };
-       });
-
-     });
```

首先获取可见元素的 ID，然后向浏览器中添加历史状态。`pushstate()`方法的第一个参数是用于后续交互的对象。我们用其保存将来用户回退到这个环节时，需显示的 tab 的 ID。例如，当用户单击 Services tab 的时候，在状态对象中会存入`#services`。

第二个参数是一个标题，用于在历史记录中进行状态标识。它与页面中的 `title` 元素没有关系，仅仅是在浏览器历史中，对历史记录项的一种标识。这里还是使用 tab 的 ID。

10.1.2 获取先前状态

在历史记录可保存的基础上，我们仍需编写代码来处理历史状态变更的情况。在用户单击后退按钮的时候，`window.onpopstate()`会被触发。我们借助它来显示存储在状态对象中的 tab。

Download html5history/javascripts/application.js

```
if(supportsHistory()){
  window.onpopstate = function(event) {
    if(event.state){
      var tab = (event.state["tab"]);
      $(".visible")
```

```
          .removeClass("visible")
          .addClass("hidden")
          .hide();
        $(tab)
          .removeClass("hidden")
          .addClass("visible")
          .show();
      }
    };
  };
```

获取 tab 的名称，使用 jQuery 通过 ID 定位需要隐藏的元素。此处隐藏和显示 tab 的代码与原先的重复了，因此需要对其重构以避免重复。

10.1.3　默认状态

在第一次浏览页面的时候，历史状态会是空，因此需要自行设置。只需要在定义 window.onpopstate( )方法的语句上方添加如下代码：

Download html5history/javascripts/application.js

```
if(supportsHistory()){
  window.history.pushState( {tab: "#welcome"}, '#welcome');
  window.onpopstate = function(event) {
    if(event.state){
      var tab = (event.state["tab"]);
      $(".visible")
        .removeClass("visible")
        .addClass("hidden")
        .hide();
      $(tab)
        .removeClass("hidden")
        .addClass("visible")
        .show();
    }
  };
};
```

现在，在浏览页面的时候，便可以通过浏览器历史来循环切换 tab 了。[1]

10.1.4　回退

此功能在 Firefox 4 和 Safari 4 中可用，并兼容 Chrome 5，但在 Internet Explorer 中不被支持。类似于 jQuery Address 插件[2]的解决方案可以提供相同的功能，但是我们不会将其作为回退方案

① 测试的时候，需要频繁地关闭浏览器并清空历史记录。有时候，这会令人感觉很麻烦。

② 参见 http://www.asual.com/jquery/address/。

来实现，因为与其称之为回退方案，不如说是带有很多附加功能的完整替代方案。请注意浏览器对操控历史记录的支持情况，因为如果在所有浏览器中都能使用这一 API 的话，我们便可以轻松地创建更具用户友好性的应用。

10.2 实例 24：跨域对话

一直以来，客户端 Web 应用脚本的交互都被限制在当前域中，无法跨域对话。这种限制机制的设计初衷是为了保护用户。[1]针对这一限制，出现了各种各样变通的实现方式，包括使用服务器端代理和特殊 URL 手段等。不过现在我们有了更好的解决办法。

HTML5 规范引入了跨文档通信（Cross-document Messaging）。通过此 API，可以让位于不同域的脚本具备相互传递消息的功能。例如，我们可以通过 http://support.awesomecompany.com 上的一个表单，向内容位于 http://www.awesomecompany.com 上的一个窗口或 `iframe` 发送内容。对于目前我们在做的项目来说，只需要这样做就够了。

AwesomeCo 新的支持页面需放置一张联系人表单，项目经理还要求在其中列出所有支持人员，并将他们的电子邮件地址置于联系人表单的旁边。支持人员名单将最终来自另一个服务器的内容管理系统，因此我们可以通过 `iframe` 将联系人列表嵌入到表单旁边。管理员会喜欢这样一个功能：用户单击联系人列表中的名字，就会在表单中自动添加其电子邮件地址。

实现方式非常简单，只不过我们需要使用 Web 服务器来进行测试。如果没有服务器的话，这份示例代码不能在所有浏览器上运行。更多内容见“简单的 Web 服务器”。

简单的 Web 服务器

如果读者觉得配置 Apache 实例比较麻烦，也不愿设置自己的服务器，那么可以使用基于 Ruby 的简单服务器（在本书的示例代码文件中可以找到）。为使 Ruby 在自己的系统中正常运行，请参考本书源代码文件中的 RUBY_README.txt。

启动服务器的方法是首先找到 `html5xdomain/contactlist`，按照如下方式运行 server.rb：

```
ruby server.rb
```

这样服务器会在 4567 端口启动。对于 `html5xdomain/supportpage` 下的 `server.rb` 也可做同样操作，在 3000 端口上启动服务器。修改 server.rb 中的配置可以修改对应的启动端口。

① 这便是所谓的同源策略。更详细的介绍见 https://developer.mozilla.org/en/Same-origin-policy-for-JavaScript。

10.2.1 联系人列表

首先我们创建联系人列表。基本的标记如下：

Download html5xdomain/contactlist/public/index.html

```
<ul id="contacts">
  <li>
    <h2>Sales</h2>
    <p class="name">James Norris</p>
    <p class="email">j.norris@awesomeco.com</p>
  </li>
  <li>
    <h2>Operations</h2>
    <p class="name">Tony Raymond</p>
    <p class="email">t.raymond@awesomeco.com</p>
  </li>
  <li>
    <h2>Accounts Payable</h2>
    <p class="name">Clark Greenwood</p>
    <p class="email">c.greenwood@awesomeco.com</p>
  </li>
  <li>
    <h2>Accounts Receivable</h2>
    <p class="name">Herbert Whitmore</p>
    <p class="email">h.whitmore@awesomeco.com</p>
  </li>
</ul>
```

在此页面上，我们还会加载 jQuery 库和自己定制的 application.js 文件，以及一个简单的样式表。将这些文件在 head 区段引入：

Download html5xdomain/contactlist/public/index.html

```
<script type="text/javascript"
   charset="utf-8"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>
<script type="text/javascript"
  src="javascripts/application.js">
</script>
<link rel="stylesheet" href="style.css" type="text/css" media="screen">
```

联系人列表的样式表如下所示：

Download html5xdomain/contactlist/public/style.css

```
ul{
  list-style: none;
}
```

```
ul h2, ul p{margin: 0;}
ul li{margin-bottom: 20px;}
```

只用了很少的一些技巧，我们便使列表看起来更整洁了。

10.2.2 发送消息

当用户在联系人列表中单击某个项时，我们会从列表项中获取电子邮件地址，然后将消息发送回父窗口。postMessage()方法带有两个参数：消息本身和目标窗口地址。下面是完整事件处理程序的代码：

Download html5xdomain/contactlist/public/javascripts/application.js

```
$(function(){
  $("#contacts li").click(function(event){
    var email = ($(this).find(".email").html());
    var origin = "http://192.168.1.244:3000/index.html";
      window.parent.postMessage(email, origin);
  });
});
```

为了匹配父窗口的 URL，读者可能需要修改代码中的 origin。[①]

接下来，需要编写一个页面来显示这个框架，并接收消息。

10.2.3 支持页面

支持页面的结构看起来会特别相似。为了不混淆，特别是考虑到该网站会放置在另一个 Web 服务器上，我们会新建一个文件夹。请确保链接了样式表、jQuery 和 application.js 文件。

该页面上需要一份联系人表单和一个 iframe。此 iframe 会指向联系人列表。代码如下：

Download html5xdomain/supportpage/public/index.html

```
<div id="form">
  <form id="supportform">
    <fieldset>
      <ol>
        <li>
          <label for="to">To</label>
          <input type="email" name="to" id="to">
        </li>
        <li>
          <label for="from">From</label>
```

① 事实并不完全如此。我们可以只用域或者甚至是一个通配符。不过这里的回退方案需要完整的 URL，而且这样做在安全性方面也比较好。

```
          <input type="text" name="from" id="from">
        </li>
        <li>
          <label for="message">Message</label>
          <textarea name="message" id="message"></textarea>
        </li>
      </ol>
      <input type="submit" value="Send!">
    </fieldset>
  </form>
</div>

<div id="contacts">
  <iframe src="http://192.168.1.244:4567/index.html"></iframe>
</div>
```

将下面的 CSS 代码添加到 style.css 中为页面应用样式：

Download html5xdomain/supportpage/public/style.css

```
#form{
  width: 400px;
  float: left;
}
#contacts{
  width: 200px;
  float: left;
}
#contacts iframe{
  border: none;
  height: 400px;
}
fieldset{
  width: 400px;
  border: none;
}

fieldset legend{
  background-color: #ddd;
  padding: 0 64px 0 2px;
}

fieldset>ol{
  list-style: none;
  padding: 0;
  margin: 2px;
}

fieldset>ol>li{
  margin: 0 0 9px 0;
```

```
  padding: 0;
}

/* 让输入保持在自己的范围内*/
fieldset input, fieldset textarea{
  display:block;
  width: 380px;
}
fieldset input[type=submit]{
  width: 390px;
}

fieldset textarea{
  height: 100px;
}
```

这份样式表将表单和 iframe 并列在一起，并对表单进行了修饰，效果见图 10-1。

图 10-1 完整的支持页面

10.2.4 接收消息

只要当前窗口接收到消息，就会触发 onmessage 事件。消息会作为 event 的属性返回。使用 jQuery 的 bind()方法注册此事件，这样在所有的浏览器中它都能正常运行。

Download html5xdomain/supportpage/public/javascripts/application.js

```
$(function(){
    $(window).bind("message",function(event){
      $("#to").val(event.originalEvent.data);
    });

});
```

jQuery 的 bind()方法封装此事件，只暴露部分属性。通过事件的 originalEvent 属性来访问可以获取我们想要的信息。

此页面可以在 Firefox、Chrome、Safari 和 Internet Explorer 8 中正常运行。接下来，我们针对 IE6 和 IE7 做些兼容性修改。

10.2.5　回退

为支持 IE6 和 IE7，我们要使用 jQuery 的 Postback 插件，此插件可以模拟跨域消息交互。使用 jQuery 的 `getScript()`方法可以在需要的时候才进行库加载。为实现这样的功能，只需要检测 `postMessage()`方法是否存在即可。

首先，修改联系人列表：

Download html5xdomain/contactlist/public/javascripts/application.js

```
if(window.postMessage){
  window.parent.postMessage(email, origin);
}else{
  $.getScript("javascripts/jquery.postmessage.js", function(){
    $.postMessage(email, origin, window.parent);
  });
}
```

jQuery 的 Postmessage 插件添加了一个 `postMessage()`方法，此方法的作用与标准的 `postMessage()`方法几乎一模一样。

现在将注意力转向支持页面。修改方式类似，添加库然后调用新添加的 `receiveMessage()`方法：

Download html5xdomain/supportpage/public/javascripts/application.js

```
if(window.postMessage){
  $(window).bind("message",function(event){
    $("#to").val(event.originalEvent.data);
  });
}else{
   $.getScript("javascripts/jquery.postmessage.js", function(){
     $.receiveMessage(
       function(event){
         $("#to").val(event.data);
       });

   });
}
```

完成了！现在我们可以在大量浏览器中跨窗口交互了。然而这只是一个开始，扩展上面的技术还可以实现双向通信。任何的窗口都可以是发送方，也都可以是接收方。所以请仔细阅读一下规范，看我们还能创建出什么应用出来！

10.3 实例 25：使用 Web Sockets 进行即时通信

多年来，Web 开发人员一直致力于研究实时交互。但是大多数实现方式局限于使用 Javascript 周期性地访问远程服务器来检测状态变更。HTTP 是一种无状态协议，因此 Web 浏览器向服务器发起连接，获取响应，最后关闭连接。要在一种无状态协议上实现实时交互，未免太过牵强。HTML5 规范引入了 Web Scockets，这使得浏览器可以向远程服务器发起有状态的连接。[①]借助 Web Sockets，我们可以构建各种强大的应用。对于 Web Sockets 的工作机制，最佳的了解方式之一是编写一个即时通信客户端。凑巧的是，AwesomeCo 的支持主页上也有这样的需求。

AwesomeCo 希望在其支持主页上建立一款基于 Web 的、简单的即时通信界面，因为其支持团队的成员位于不同的城市，这样一来他们就可以进行内部交流了。我们将使用 Web Sockets 来实现即时通信服务器的 Web 界面。用户可以连接并向服务器发送信息。只要联机，用户都能接收到该消息。网站的访问者可以通过发送类似“/昵称”的信息为自己注册昵称，这仿照了 IRC 的聊天协议。我们不需要完完整整地编写这类服务器，因为有幸已经有别的开发人员写好了。[②]

10.3.1 即时通信界面

我们会按照图 10-2 所示的效果建立一个非常简单的即时通信界面。其中有一个修改用户昵称的表单、一个用于显示消息的大文本区域、以及一个在聊天中用来发送消息的表单。

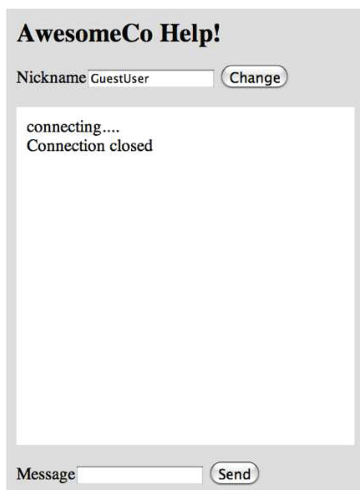


图 10-2 即时通信界面

① Web Sockets 也已经发展形成了独立的规范，见 http://www.w3.org/TR/websockets/。

② 关于这种服务器的更多内容，请参见 10.3.5 节。

在全新的 HTML5 页面中，我们为即时通信界面添加标记，包括两个表单和一个用于包含聊天信息的 div。

Download html5_websockets/public/index.html

```
<div id="chat_wrapper">
  <h2>AwesomeCo Help!</h2>
  <form id="nick_form" action="#" method="post" accept-charset="utf-8">
    <p>
      <label>Nickname
        <input id="nickname" type="text" value="GuestUser"/>
      </label>
      <input type="submit" value="Change">
    </p>
  </form>

  <div id="chat">connecting....</div>

  <form id="chat_form" action="#" method="post" accept-charset="utf-8">
    <p>
      <label>Message
        <input id="message" type="text" />
      </label>
      <input type="submit" value="Send">
    </p>
  </form>
</div>
```

我们还需要添加到样式表的链接，以及包含与 Web Sockets 服务器交互的代码的 JavaScript 文件。

Download html5_websockets/public/index.html

```
<script src='chat.js' type='text/javascript'></script>
<link rel="stylesheet" href="style.css" media="screen">
```

此处的样式表包含下列样式定义：

Download html5_websockets/public/style.css

```
Line 1  #chat_wrapper{
   -      width: 320px;
   -      height: 440px;
   -      background-color: #ddd;
   5      padding: 10px;
   -    }
   -    #chat_wrapper h2{
   -      margin: 0;
   -    }
  10
   -    #chat{
```

```
    width: 300px;
    height: 300px;
    overflow: auto;
15  background-color: #fff;
    padding: 10px;
  }
```

我们在第 14 行设定了聊天消息文本区域的 overflow 属性，这样即指定了其高度，又可将不能容纳的文本都隐藏起来，并使之可通过滚动条查看。

界面做好了以后，我们就可以开始编写用于与即时通信服务器交互的 JavaScript 代码了。

10.3.2 与服务器交互

不论使用的是哪种 Web Sockets 服务器，模式都是通用的。我们会向服务器发起一个连接，然后监听服务器发回的事件并做适当响应。

事　件	描　述
onopen()	与服务器的连接建立后触发
onmessage()	与服务器的连接发送消息时触发
onclose()	与服务器的连接丢失或者关闭时触发

在 chat.js 文件中，首先需要按如下方式连接 Web Sockets 服务器：

Download html5_websockets/public/chat.js

```
var webSocket = new WebSocket('ws://localhost:9394/');
```

与服务器连接成功后，应该通知用户。按如下方式定义 onopen()方法：

Download html5_websockets/public/chat.js

```
webSocket.onopen = function(event){
  $('#chat').append('<br>Connected to the server');
};
```

当浏览器打开与服务器的连接后，聊天窗口中会显示一条消息。接着，我们需要显示发送到聊天服务器的消息。按如下方式定义 onmessage()方法：

Download html5_websockets/public/chat.js

```
webSocket.onmessage = function(event){
  $('#chat').append("<br>" + event.data);
  $('#chat').animate({scrollTop: $('#chat').height()});
};
```

消息通过 event 对象的 data 属性返回，我们只需要将其添加到聊天窗口中即可。这里使用前置换行来规范响应消息，让其各自独立成行。当然开发人员可以选择自己喜欢的任何标记方式。

接下来，我们要对关闭连接进行处理。一旦连接关闭，onclose()方法就会被触发。

Download html5_websockets/public/chat.js

```
webSocket.onclose = function(event){
  $("#chat").append('<br>Connection closed');
};
```

现在需要对聊天表单关联文本区域，以便可以发送消息至即时通信服务器。

Download html5_websockets/public/chat.js

```
$(function(){
  $("form#chat_form").submit(function(e){
     e.preventDefault();
     var textfield = $("#message");
     webSocket.send(textfield.val());
     textfield.val("");
   });
})
```

在表单提交事件中，抓取表单域的值，使用 send()方法将其发送到即时通信服务器。

我们用同样的方式来实现修改昵称的功能，只不过要在消息前面附加“/昵称”，这样服务器就会识别并修改用户的昵称。

Download html5_websockets/public/chat.js

```
$("form#nick_form").submit(function(e){
  e.preventDefault();
  var textfield = $("#nickname");
  webSocket.send("/nick " + textfield.val());
});
```

大功告成。Safari 5 和 Chrome 5 的用户可以使用此客户端立即加入到实时通信中。当然我们仍然需要为非原生支持 Web Sockets 的浏览器提供支持。在此，我们使用 Flash 做替代。

10.3.3　回退

不一定所有的浏览器都支持套接字连接，但是 Adobe Flash 已经为其提供支持很久了。我们可以将 Flash 作为套接字通信层，感谢 web-socket-js[①]库，有了它实现 Flash 回退方案就是小菜一碟了。

下载插件[②]并放置到我们的项目中。然后需要往页面中引入 3 个 Javascript 文件：

① 参见 http://github.com/gimite/web-socket-js/。

② 参见 http://github.com/gimite/web-socket-js/archives/master。

Download html5_websockets/public/index.html

```
  <script type="text/javascript" src="websocket_js/swfobject.js"></script>
  <script type="text/javascript" src="websocket_js/FABridge.js"></script>
  <script type="text/javascript" src="websocket_js/web_socket.js"></script>

  <script src='chat.js' type='text/javascript'></script>
  <link rel="stylesheet" href="style.css" media="screen">

</head>
<body>
<div id="chat_wrapper">
  <h2>AwesomeCo Help!</h2>
  <form id="nick_form" action="#" method="post" accept-charset="utf-8">
    <p>
      <label>Nickname
        <input id="nickname" type="text" value="GuestUser"/>
      </label>
       <input type="submit" value="Change">
     </p>
   </form>

   <div id="chat">connecting....</div>

   <form id="chat_form" action="#" method="post" accept-charset="utf-8">
     <p>
       <label>Message
         <input id="message" type="text" />
       </label>
       <input type="submit" value="Send">
     </p>
   </form>
 </div>

 </body>
 </html>
```

唯一要修改 chat.js 文件的地方是，设置一个变量，用来指定 WebSocketMain 文件的路径：

Download html5_websockets/public/chat.js

```
WEB_SOCKET_SWF_LOCATION = "websocket_js/WebSocketMain.swf";
```

这样，若即时通信服务器也提供 Flash 套接字策略（Socket Policy）文件，我们的应用便可以运行在所有主流浏览器中了。

10.3.4 什么是 Flash 套接字策略

出于安全方面的考虑，Flash Player 只能通过套接字同允许连接到 Flash Player 的服务器通信。

Flash Player 会首先尝试从 843 端口获取 Flash 套接字策略文件，然后是服务器使用的端口。它期望从服务器获取到的响应如下：

```
<cross-domain-policy>
     <allow-access-from domain="*" to-ports="*" />
</cross-domain-policy>
```

这是一份很常规的策略文件，允许所有人连接到此服务。如果我们应对的是更敏感的数据，那可能就需要指定更严格的策略了。需要注意的是，我们需要将其放置在运行 Web Sockets 的服务器上，端口号可以同 Web Sockets 的相同，也可使用 843。

这一区段的示例代码包含一个简单的 Flash 套接字策略服务器，它是用 Ruby 写的，供我们在测试时使用。在 10.3.5 节可以看到如何配置测试环境。

这里的即时通信服务器只是触及了 Web Sockets 的皮毛。有了 Web Sockets，我们便有了一种健壮而简洁的方式来将数据推送到访问者的浏览器中。

10.3.5 服务器

本书的源码中包含一个版本的 Web Sockets 服务器。因为它是用 Ruby 编写的，所以我们需要 Ruby 解释器。对于如何搭建 Ruby 运行环境，参见本书源代码文件中的 RUBY_README.txt。

启动方式是找到相应路径，输入下述代码：

```
ruby server.rb
```

除了即时通信服务器之外，还有两种其他的服务器可能会在测试本章代码时用到。第一个服务器是 `client.rb`,服务于即时通信前端和 Javascript 文件。另一个服务器是 `flashpolicyserver`，服务于 Flash 策略文件，该文件为基于 Flash 的 Web Sockets 回退代码所用，用于与实际的即时通信服务器建立连接。Flash Player 使用这些策略文件来判定其是否可以与远程域进行对话。

如果我们使用的是 Mac 或者基于 Linux 的操作系统，可以从 html5_websockets 路径下一次性启动所有的服务器：

```
rake start
```

10.4 实例 26： Geolocation

Geolocation 是一种基于用户电脑位置发现用户所在位置的技术。当然这里的“电脑”不仅包括台式机，同样还指代智能手机、平板电脑，以及其他便携式设备。Geolocation 通过用户电脑的 IP 地址、MAC 地址、Wi-Fi 热点位置，甚至 GPS 坐标来判断用户的所属位置。虽然不是严格作

为 HTML5 规范的一部分，但由于是同期产生的，因此通常与 HTML5 技术配合使用。与 Web Storage 不同，Geolocation 从来都没有作为 HTML5 规范的一部分出现过。与 Web Storage 相同的是，Geolocation 是一项非常有用的技术，并已被 Firefox、Safari 以及 Chrome 实现。让我们来看看如何使用 Geolocation。

10.4.1 定位 Awesomeness

前面我们为 AwesomeCo 的 Web 站点建立了一个联系人页面。CIO 又提出新需求，看是否可以在地图上除显示各 AwesomeCo 支持中心之外再显示用户的位置。他希望看到一个原型，因此我们将迅速搭建一套系统并使之能马上运行。

我们选择使用 Google 的 Static Map API，这是因为一方面它不需要 API 序列号，另一方面是我们只需用到一个非常简单的地图。

AwesomeCo 服务中心位置不一，分别是俄勒冈州的波特兰、伊利诺伊州的芝加歌，以及罗得岛州的普罗维登斯。使用 Google 的 Static Map API 可以很容易地在地图上标记这些点。我们需要做的只是建立 `img` 标签，将 URL 地址传进去。如下所示：

Download html5geo/index.html

```
<img id="map" alt="Map of AwesomeCo Service Center locations"
src="http://maps.google.com/maps/api/staticmap?
&amp;size=900x300
&amp;sensor=false
&amp;maptype=roadmap
&amp;markers=color:green|label:A|1+Davol+square,+Providence,+RI+02906-3810
&amp;markers=color:green|label:B|22+Southwest+3rd+Avenue,Portland,+OR
&amp;markers=color:green|label:C|77+West+Wacker+Drive+Chicago+IL">
```

首先定义了图片的大小，然后告诉 Maps API 我们没有使用任何传感器设备（包括 GPS 或者客户端地理定位设备）以及要传给这个地图的信息。然后通过指定标签和路径的方式来定义地图中的各个标记。如果有标记的话，我们可以对其应用逗号分隔的经纬度值，但这里这样做目的是方便演示。

10.4.2 如何定位

我们需要在地图中标注访问者的当前位置。这里使用经度和纬度组成的新标记来实现。我们可以要求浏览器抓取访问者的经纬度，如下所示：

Download html5geo/index.html

```
navigator.geolocation.getCurrentPosition(function(position) {
```

```
    showLocation(position.coords.latitude, position.coords.longitude);
  });
```

这个方法会弹出窗口要求用户提供坐标。如果用户允许我们使用其位置信息，就调用 showLocation()方法。

showLocation()方法接收经度和纬度信息，重建图片并替代已有 image 的源。下面是此方法的实现方式：

Download html5geo/index.html

```
Line 1  var showLocation = function(lat, lng){
   2      var fragment = "&markers=color:red|color:red|label:Y|" + lat + "," + lng;
   3      var image = $("#map");
   4      var source = image.attr("src") + fragment;
   5      source = source.replace("sensor=false", "sensor=true");
   6      image.attr("src", source);
   7  };
```

无需重复整个图片源代码，为现有的图片源添加经度和纬度信息即可。

在将修改后的图片源设置回文档之前，需要将 sensor 参数由 false 设置为 true，这在第 5 行的 replace()方法中实现。

在浏览器中试运行，可以看到我们自己的位置，使用“Y”标记，见图 10-3。



图 10-3 使用“Y”将我们当前的位置标记在了地图上

10.4.3 回退

现在为止，页面的访问者仍能看到带有 AwesomeCo 支持中心位置的地图，但如果尝试加载

页面的话，会产生一个 JavaScript 错误。所以在试图获取用户位置之前，我们需要先检测对地理位置定位的浏览器支持情况，如下所示：

Download html5geo/index.html

```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(function(position) {
    showLocation(position.coords.latitude, position.coords.longitude);
  });
}else{
};
```

Google 的 Ajax API[①]有查看位置的功能，因此它是一款理想的回退解决方案。如果要将其正式部署在我们自己的网站上，需要获得 API 授权码，但对于本地调试来说是不需要的。[②]

回退代码如下所示：

Download html5geo/index.html

```
Line 1  var key = "your_key";
     -  var script = "http://www.google.com/jsapi?key=" + key;
     -  $.getScript(script, function(){
     -    if ((typeof google == 'object') &&
     5       google.loader && google.loader.ClientLocation) {
     -         showLocation(google.loader.ClientLocation.latitude,
     -                      google.loader.ClientLocation.longitude);
     -     }else{
     -       var message = $("<p>Couldn't find your address.</p>");
    10       message.insertAfter("#map");
     -     };
     -  });
```

这里使用了 jQuery 的 `getScript()`方法来加载 Google Ajax API。然后第 5 行使用 Google 的 `ClientLocation()`方法获取用户位置并触发 `showLocation()`方法将位置标记在地图上。

不过，不是所有的 IP 地址都能被 Google 定位到，所以我们可能还是无法将某些用户标记在地图上。因此第 9 行，我们在图片的下方放置了一条信息。这个回退方案尽管不是那么无懈可击，但却给我们提供了定位用户的更大机会。

对于从客户端获取位置坐标来说，在没有可靠方法的情况下，我们需要提供一种让用户自己提供地址的方式，不过这项工作就留给读者去完成吧。

---

① 参见 http://code.google.com/apis/ajax/documentation/#ClientLocation。

② 如果使用 http://localhost/来访问的话也需要授权码。授权码可在 http://code.google.com/apis/ajaxsearch/signup.html 获取到。

10.4.4 未来展望

本章所讨论的技术虽然不都是 HTML5 规范的一部分，但却代表着 Web 开发的未来。未来的趋势是将越来越多的东西推送至客户端。更好的历史管理将使得 Ajax 和客户端应用变得更加直观。Web Sockets 可以替代周期性轮询远程服务器的方式来展现实时数据。跨文档消息机制可以让我们归并之前完全没有可能发生交互的 Web 应用。Geolocation 最终会辅助我们建立更好的位置感知 Web 应用，随着移动计算市场的增长，这部分应用会变得越来越有意义。

建议研究这些 API，并注意其适用性。不久的将来，你会发现它们变成我们 Web 开发工具中的“重量级武器”。

第 11 章 未来的发展方向

本书大部分篇幅着眼于当下可用的技术，其实还有一些技术很快就能具备实用条件，继而会让我们基于规范的 Web 开发变得更加有趣，如带有 WebGL 的 3D 画布支持、新的 storage API、CSS3 变换以及原生拖放支持等。本章讨论的就是这些即将成型的技术，这样我们便能有个期望值。我们即将讨论的这些技术至少可以应用在一款浏览器中，但却没有足够好的回退方案，或者暂时还不足以拿来使用，如下所示。

- CSS3 变换——交互中的动画。[C3、S3.2、F4、O10.5、IOS3.2、A2]
- Web Workers——为 JavaScript 提供后台处理技术。[C3、S4、F3.5、O10.6]
- 带有 WebGL 的 3D 画布——在画布中创建 3D 对象。[C5、F4]
- IndexedDB——同 NoSQL 解决方案类似的高级客户端键/值数据库存储技术。[F4]
- 拖放——拖放操作的 API。[C3、S4、F3.5、IE6、A2]
- 表单验证——输入内容的客户端验证技术。[C5、S5、10.6]

我们从 CSS3 变换入手，看如何在 WebKit 浏览器中使用。

11.1 CSS3 变换

在好的用户体验设计中，邀请式交互是非常重要的。CSS 已经实现了对`:hover`伪类的支持，这样我们可以在元素中添加一些基本的交互提示。下面的 CSS 标记可以让链接变得更像按钮：

Download css3transitions/style.css

```
a.button{
  padding: 10px;
  border: 1px solid #000;
  text-decoration: none;
}
a.button:hover{
 background-color: #bbb;
 color: #fff
}
```

当鼠标移动到按钮上方时，背景色由白变灰，同时字体颜色由黑变白。这是一种即时变换。CSS3 变换[①]为我们带来了更多选择，包括只需 JavaScript 即可产生的简单动画。例如，我们可以将上面的这种变换改为淡入淡出，方法是将下面特别标出的代码添加到样式定义中：

Download css3transitions/style.css

```
Line 1  a.button{
   -      padding: 10px;
   -      border: 1px solid #000;
   -      text-decoration: none;
   5      -webkit-transition-property: background-color, color;
   -      -webkit-transition-duration: 1s;
   -      -webkit-transition-timing-function: ease-out;
   -    }
   -
  10
   -    a.button:hover{
   -     background-color: #bbb;
   -     color: #fff
   -    }
```

在第 5 行，我们指定了要应用变换的属性。在这里改变的是背景色和前景色。第 6 行定义了动画时长。第 7 行指定了变换的时间函数。

时间函数

transition-timing-function 属性描述的是前面设定的动画时长内，变换发生的方式。我们使用三次贝塞尔曲线定义此时间函数，三次贝塞尔曲线是通过一幅图上的 4 个控制点来定义的。每个点都有 X 值和 Y 值，取值范围是 0～1。第一个和最后一个控制点始终为(0.0,0.0)和(1.0,1.0)，中间两个点决定了曲线的形状。

线性曲线的控制点是两个端点，会形成 45 度角的直线。线性曲线的 4 个控制点分别为：(0.0, 0.0)、(0.0,0.0)、(1.0, 1.0)、(1.0, 1.0)，效果如图 11-1 所示。

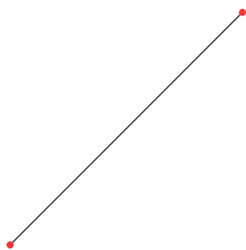


图　11-1

① 参见 http://dev.w3.org/csswg/css3-transitions/。

ease-in 曲线稍复杂一些，其控制点为 (0.0, 0.0)、(0.42,0.0)、(1.0, 1.0)、(1.0,1.0)，效果如图 11-2 所示。

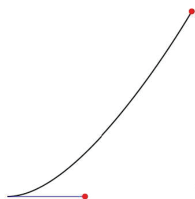


图 11-2

同线性曲线相比，ease-in 曲线只有第二个点不同，正因为这个点，才让直线的左下部变成了曲线。

ease-in-out 曲线更复杂，在底部和顶部都有曲线，如图 11-3 所示。该曲线对应的控制点为 (0.0, 0.0)、(0.42,0.0)、(0.58, 1.0)、(1.0,1.0)。

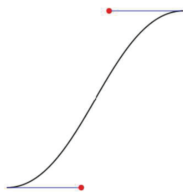


图 11-3

我们可以直接在 CSS 的属性中设置这些控制点，也可以像上面例子中一样，使用预设的效果。

可供选择的效果有 `default`、`ease-in`、`ease-out`、`ease-in-out`、`ease-out-in` 以及 `cubic-bezier`，其中 `cubic-bezier` 允许开发人员自行设定控制点。

如果希望使用恒定的比率，请选择 `linear`。如果希望动画开始很慢并逐渐变快，请选择 `ease-in`。对于创建曲线方面的更多知识，读者可以在一个很强大的公开域脚本[1]中找到很多例子，并能在它的帮助下查看坐标。

玩转变换的时候，有一点要铭记在心，那就是界面实用性第一绚丽第二。千万不要创建会干扰用户的变换，如忽隐忽现的东西或者太长的动画。CSS3 动画[2]也是一种会引发我们兴趣的技术，它是另一种基于时间修改 CSS 属性的方法。

① 参见 http://www.netzgesta.de/dev/cubic-bezier-timing-function.html。

② 参见 http://www.w3.org/TR/css3-animations/。

11.2　Web Workers

Web Workers[①]并不是 HTML5 规范的一部分，但如果我们要做一些客户端的后台处理，就会发现这项技术其实非常有用，因此在这里有必要提及。

所有的客户端编程只用到 JavaScript，但 JavaScript 是单线程语言，也就是说同一时间只能处理一件事情。如果一项任务耗时过长，那么用户就必须等待，直到该任务结束。Web Workers 通过一种简单的写入并发程序的方式，很好地解决了这个问题。

假设有一份名为 worker.js 的脚本是用来处理图片的，那么可以这样调用：

Download webworkers/application.js

```
var worker = new Worker("worker.js");
```

任何 JavaScript 文件都可以作为 worker 被调用，但为了保证 worker 的非耦合性，所有 worker 脚本都不允许访问 DOM。这就意味着我们无法直接操控页面元素。

主脚本可以使用 postMessage()方法发送到 worker 脚本的消息，如下所示：

Download webworkers/application.js

```
$("#button").click(function(event){
  $("#output").html("starting...");
  worker.postMessage("start");
});
```

然后 worker 脚本可以向主页面回送消息，使用的还是 `postMessage()`方法：

Download webworkers/worker.js

```
onmessage = function(event) {
  if(event.data === "start"){
    // 循环内可替换为自己希望的操作
    for (var x = 1; x <= 100000; x++){
       postMessage(x);
    }
  }
};
```

我们通过监听主脚本中的 `onmessage` 事件来响应这些事件。每当 worker 返回消息，就会触发下面的代码：

Download webworkers/application.js

```
worker.onmessage = function(event){
  $("#output").html(event.data);
}
```

---

① 参见 http://www.whatwg.org/specs/web-workers/current-work/。

此 API 跟跨域消息（参见 10.2 节）的 API 类似。Internet Explorer 不支持 Web Workers，因此我们需要借助 Google Chrome Frame。不过如果我们要在客户端做一些繁重的非阻塞事务，那么敬请留意 Web Workers 以后的发展。

11.3 原生拖放支持

一段时间以来，我们借助 JavaScript 库已经可以实现用户拖放界面元素的类似功能了。不过 W3C 将 Microsoft 的拖放实现纳入了 HTML5 规范。[①]支持原生拖拽的浏览器包括 Firefox、Safari、Internet Explorer 和 Chrome，但事实上还是比较混乱的。

最初的实现方式比较直观。首先指定某元素是可拖拽的，然后指定另外一个元素来等待对象释放，有对象释放后就执行一段代码。

事实上，并非如此简单。为演示起见，我们创建一个简单的拖放界面，允许通过将小图拖放到指定区域来加载大图。

Download **html5drag/index.html**

```
<div id="images">
  <img src="images/red_thumb.jpg"
       data-large="images/red.jpg" alt="A red flower">
  <img src="images/purple_thumb.jpg"
       data-large="images/purple.jpg" alt="A white and purple flower">
  <img src="images/white_thumb.jpg"
       data-large="images/white.jpg" alt="A white flower">

</div>

<div id="preview">
  <p>Drop images here</p>
</div>
```

这里使用了用户自定义数据属性来保存大版本的图片。

接下来，添加一些基本样式，为两列添加浮动效果：

Download **html5drag/style.css**

```
#images img{
  -webkit-user-drag
}

#images{
  float: left;
```

① 参见 http://dev.w3.org/html5/spec/dnd.html#dnd。

```
  width: 240px;
  margin-right: 10px;
}

#preview{
  float: left;
  width: 500px;
  background-color: #ddd;
  height: 335px;
}

.hover{
  border: 10px solid #000;
  background-color: #bbb !important;
}
```

至此，界面效果如图 11-4 所示。接下来添加一些事件，以便拖曳图片。

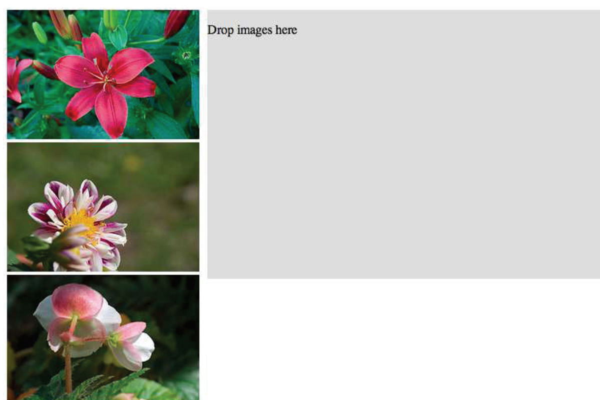


图 11-4　图片查看器

11.3.1　拖放事件

拖放元素涉及几个事件。

事　件	描　述
`ondragstart`	在用户开始拖动对象时触发
`ondragend`	不论何种原因，只要用户停止拖动对象就触发
`ondragenter`	可拖动对象移动到释放区域时触发
`ondragover`	用户将元素拖到释放区域时触发
`ondragleave`	用户将元素拖出释放区域时触发
`ondrop`	用户成功将元素拖入释放区域并释放时触发
`ondrag`	用户拖动元素时触发，持续发生但可指定鼠标光标的*X*和*Y*坐标

这是处理拖放所用到的全部 7 种事件，其中一部分事件带有默认行为。如果不对其重写，会引发错误。

首先，我们需要将所有列表项都设置为可拖曳的：

Download html5drag/application.js

```
var contacts = $('#images img');
contacts.attr('draggable', 'true');
```

我们在此添加了 draggable 的 HTML5 属性。我们可以在页面标记中完成此事，但考虑到要让 JavaScript 来处理交互，所以将属性应用到了脚本中。

当拖动图片的时候，要获取大图的地址并保存下来。我们为此绑定 ondragstart 事件，为了保持其简洁性和跨平台性，这里使用 jQuery 的 bind()方法[①]。

Download html5drag/application.js

```
Line 1 contacts.bind('dragstart', function(event) {
  2        var data = event.originalEvent.dataTransfer;
  3        var src = $(this).attr("data-large");
  4        data.setData("Text", src);
  5        return true;
  6    });
```

规范中提供了一种名为 dataStorage 的机制，可以让我们指定数据类型和定义数据，而数据会被当成事件的一部分一并发送出去。jQuery 的 bind()方法用自身对象包装事件，因此在第 2 行我们使用 originalevent 属性来访问实际事件。第 4 行，我们在事件中使用 setData()方法为图片保存 URL，数据类型为 Text。

现在便可以拖动元素了。释放元素时如何触发事件呢？让我们继续讨论。

11.3.2 释放元素

我们将“To”表单作为被拖曳元素的落入对象，因此要找到这一表单并绑定 drop 事件。

Download html5drag/application.js

```
Line 1 var target = $('#preview');
  -
  -    target.bind('drop', function(event) {
  -      var data = event.originalEvent.dataTransfer;
  5      var src = ( data.getData('Text') );
  -
  -      var img = $("<img></img>").attr("src", src);
```

① 注意，使用此方法时，我们在这些事件前面删除了 on 前缀。

```
-     $(this).html(img);
-     if (event.preventDefault) event.preventDefault();
10    return(false);
-   });
```

首先，在第 5 行我们使用 getData()方法获取随事件一起传送的图片地址。然后创建一个新的 img 元素，放入内容区域。

为了避免在目标上释放被拖曳元素时触发默认事件，我们需要取消默认的 ondrop()事件。操作方法是同时使用 preventDefault()和 return false。return false 是 Internet Explorer 需要的，而其他浏览器需要 preventDefault()。

现有的代码直接放在 Chrome 或 Safari 中是无法完全正常运行的。至少，我们还需要重写 ondragover 元素。否则，ondrag 事件是无法响应的。因此按照如下代码进行修改：

Download html5drag/application.js

```
target.bind('dragover', function(event) {
  if (event.preventDefault) event.preventDefault();
  return false;
});
```

这里同样要取消默认事件，操作方式跟上面的 ondrop 事件类似。所以参考如下方式修改 ondragend 事件：

Download html5drag/application.js

```
contacts.bind('dragend', function(event) {
  if (event.preventDefault) event.preventDefault();
  return false;
});
```

这段代码的作用是，取消停止拖动元素时将触发的所有浏览器事件，但先前定义的 ondrop 事件不受影响。

11.3.3　修改样式

当用户将某元素拖曳到某个可释放区域时，我们希望提醒用户知道。使用 ondragenter 和 ondragleave 方法可以做到：

Download html5drag/application.js

```
target.bind('dragenter', function(event) {
  $(this).addClass('hover');
  if (event.preventDefault) event.preventDefault();
  return false;
});
```

```
 target.bind('dragleave', function(event) {
   $(this).removeClass('hover');
   if (event.preventDefault) event.preventDefault();
   return false;
});
```

上述代码应用到样式表中的 hover 类，会在这些事件触发的时候被应用和移除。

11.3.4 拖动文件

在页面上拖动文本和元素仅仅是初级阶段，规范还允许开发人员创建这样的界面，即可接收来自客户端电脑的文件。上传图片或者附件就像拖动文件到特定区域一样简单。实际上，在 Firefox 3.6 和 Chrome 5 中使用 Google 的 Gmail，会发现已经有这个接收功能了。

关于拖放的更多内容，建议阅读 Leslie Michael Orchard 编写的专栏[①]。

11.3.5 并不完美

不同浏览器中拖放行为的表现不尽一致。IE8 虽然支持拖放，但如果我们将 setData()的数据类型从 Text 改为 Url 的话，就不行了。

此外，在 Safari 4 中为了支持对图片和链接之外的其他元素的拖动，需要在样式表中添加额外的 CSS：

```
#contents li{
  -webkit-user-drag
}
```

在全书中，我们多次探讨了从内容中把样式和动作分离出来是多么重要。这段代码恰恰符合这样的概念。

不要将文本拖到表单域中。主流浏览器已经实现了此功能，但并没有较好的方式来重写该动作。

借助诸如 jQuery UI[②]之类支持拖放的 JavaScript 库，我们可以使用更少的代码获得更佳的效果。

即使使用了库，我们仍会面临一个问题——可访问性。对于无法使用鼠标的用户，规范中没有提到任何替代办法。如果在界面中实现了拖放功能，那么还需要实现一个辅助功能，即无需 JavaScript 或者鼠标也可正常运行的方法，而此方法也依赖于实际需求。

规范的潜力很大，但确实还有一些事情有待解决。因此需要提醒大家注意的是，仅应在必要

---

① 参见 http://decafbad.com/blog/2009/07/15/html5-drag-and-drop。

② 参见 http://docs.jquery.com/UI/Draggable。

的时候使用，不要逼迫用户使用他们不具备的东西。

11.4 WebGL

本书讨论过 canvas 元素的 2D 上下文，不过一份关于如何使用 3D 对象的规范正在制定中。WebGL[①]规范不是 HTML5 的一部分，但 Apple、Google、Opera 和 Mozilla 都是这一工作组的成员，并在各自的浏览器中对其实现了相应的支持。

3D 图像方面的内容远远超出了本书的讲述范围，不过在 Learning WebGL[②]网站上有很多非常好的示例和教程。

11.5 Indexed Database API

本书中讨论到两种在客户端存储数据的方式：Web Storage 和 Web SQL Storage。Mozilla 基金会认为 Web SQL 规范存在问题，指出不应该将这一规范建立在特定的 SQL 引擎基础之上。于是他们引入了一种新的规范——Indexed Database [③]API，并计划将其列为自己的规范。

同 Web Storage API 的 localStorage 和 sessionStorage 类似，Indexed Database API 也是一种键/值对存储，但不同的是，后者提供更高级的查询功能。不过，在本书编写之时，还没有浏览器实现此规范，因此无需深入讨论任何实现细节，因为很可能在其实现的时候规范已经发生变化了。Firefox 4 和 Chrome 7 有可能会支持此规范。

Indexed Database API 是需要我们重点关注的规范，因为 Web SQL 现在陷入了僵局。Mozilla 已经多次声明不会在 Firefox 中实现 Web SQL。这是因为 Mozilla 不习惯 SQL 语法，他们认为 Web SQL 规范不应该建立在某个特定的数据库实现上。Web SQL 规范使用的是 SQLite 数据库语法，可能会影响这一规范的独立性。不过，Internet Explorer 很可能会将其实现，因为 Microsoft 在其开发中显示出了对 Web SQL 的极大兴趣。[④]

11.6 客户端表单验证

在客户端表单验证方面，HTML5 规范列出了一些属性供我们使用。这样，在用户将数据提交至服务器之前，我们便可捕获简单的输入错误。同样的功能，使用 JavaScript 的实现已经存在多年了，不过 HTML5 表单可以使用新属性来指定动作。

---

① 参见 https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/doc/spec/WebGL-spec.html。

② 参见 http://learningwebgl.com/blog/?p=11。

③ 参见 http://www.w3.org/TR/IndexedDB/。

④ 参见 http://hacks.mozilla.org/2010/06/beyond-html5-database-apis-and-the-road-to-indexeddb/。

通过添加 required 属性可以确保用户必填某域：

Download html5validation/index.html

```
<label for="name">Name</label>
<input type="text" name="name" autofocus required id="name">
```

这样浏览器便会阻止表单提交，并显示一条贴心的错误信息，而我们则一条 JavaScript 验证语句都不用编写。Opera 已经支持此功能，见图 11-5。

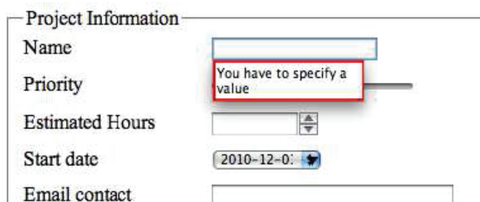


图 11-5 Opera 显示一条高亮的警告

这样做节省了用户时间，使他们无需等待从服务器返回的响应即可发现错误。此功能可能被禁用或者不被支持或者只是未被正确实现，所以我们仍需建立服务器端的数据验证机制。这绝对是当前应该考虑的问题。因为只有这样，我们才能轻松地定位必填域，并使用 CSS 样式化界面显示，以便突显必填域。

更进一步控制用户输入的话，可以使用 pattern 属性。该属性通过指定一个正则表达式来确保用户输入的内容满足我们预期。

Download html5validation/index.html

```
<input type="text"
       name="acctnumber" id="acctnumber"
       required
       pattern="^[1-9]+[0-9]*$">
```

尽管目前没有一款浏览器在用户界面中使用此功能，但是使用此标记作为 JavaScript 验证库的基本验证功能还是比较容易实现的。

11.7 前进！

作为一名开发人员，激动人心的时刻到了。对于 Web 开发人员面临的未来，本书所讲述的还只是皮毛。关于规范需了解的东西还有很多，在此希望大家深入挖掘。希望大家基于在本书学到的知识，继续探索，并在此过程中留意各种各样的规范。

现在，去建立令人叹为观止的应用吧！

附录 A

功能快速索引

在后续的描述中，浏览器支持情况使用方括号加代码缩写和最低支持版本号的形式来表示。缩写代码所表示的意义分别是：C 表示 Google Chrome，F 表示 Firefox，IE 表示 Internet Explorer，O 表示 Opera，S 表示 Safari，IOS 表示使用移动版 Safari 的 iOS 设备，A 表示 Android 浏览器。

A.1 新元素

下面是在 2.1 节引入的新元素。

- <header>——定义页面或者区段的页眉。[C5、F3.6、IE8、S4、O10]
- <footer>——定义页面或者区段的尾部。[C5、F3.6、IE8、S4、O10]
- <nav>——定义页面或者区段的导航链接部分。[C5、 F3.6、IE8、S4、O10]
- <section>——定义页面或者一组内容的逻辑区域。[C5、F3.6、IE8、S4、O10]
- <article>——定义一篇文章或整段内容。[C5、F3.6、IE8、S4、O10]
- <aside>——定义次要或相关内容。[C5、F3.6、IE8、S4、O10]
- <meter>——描述度量衡。[C5、F3.5、S4、O10]
- <progress>——显示实时目标进度的控件。[本书出版时尚无浏览器对其提供支持]

A.2 属性

- 自定义数据属性——允许使用 data-前缀为已有的任意元素添加自定义属性（引自 2.2 节）。[所有的浏览器都可通过 JavaScript 的 getAttribute()方法读取这些属性。]
- 在位编辑支持 <p contenteditable>lorem ipsum</p>]——通过浏览器提供对在位内容编辑的支持（引自 3.4 节）。[C4、S3.2、IE6、O10.1]

A.3 表单

下面是在 3.1 节引入的内容。

- 电子邮件输入域 [<input type="email">]——显示用于输入电子邮件地址的表单域。[O10.1、IOS]
- URL 输入域 [<input type="url">]——显示用于输入 URL 的表单域。[O10.1、IOS]
- 电话号码输入域 [<input type="tel">]——显示用于输入电话号码的表单域。[O10.1、IOS]
- 搜索域 [<input type="search">——显示用于输入搜索关键字的表单域。[C5、S4、O10.1、IOS]
- 滑块（范围选择）[<input type="range">]——显示滑块控件。[C5、S4、O10.1]
- 数值设定框 [<input type="number">]——显示用于输入数字的表单域，通常以选值框形式显示。[C5、S5、O10.1、IOS]
- 日期选择域 [<input type="date">]——显示用于选择日期的表单域，支持 date、month 以及 week。[C5、S5、O10.1]
- 可选择时间的日期选择域 [<input type="datetime">]——显示用于选择时间和日期的表单域，支持 datatime、datetime-local 或 time。[C5、S5、O10.1]
- 颜色选择域 [<input type="color">]——显示取色器的表单域。（Chrome5 和 Safari5 虽然可以识别颜色选择域，但不显示任何控件。）[C5、S5]

A.4　表单域属性

- 自动聚焦功能支持 [<input type="text" autofocus>]——支持为特定表单元素设定焦点（引自 3.2 节）。[C5、S4]
- 占位文本支持 [<input type="email" placeholder="me@example.com">]——支持在表单域内显示占位文本（引自 3.3 节）。[C5、S4、F4]
- required [<input type="email" required >]——设定必填域（引自 11.6 节）。[C5、S5、O10.6]
- pattern [<input type="text" pattern="^[1-9]+[0-9]*$">]——以正则表达式模式匹配方式验证表单域的数据（引自 11.6 节）。[C5、S5、O10.6]

A.5　可访问性

- role 属性 [<div role="document">]——为屏幕阅读器识别元素的作用（引自 5.1 节）。[C3、F3.6、S4、IE8、O9.6]
- aria-live [<div aria-live="polite">]——识别可自动更新的区域，可能要通过 Ajax 完成（引自 5.2 节）。[F3.6（Windows）、S4、IE8]

- aria-atomic [<div aria-live="polite" aria-atomic="true">]——识别需要阅读的内容是实时区域的整篇内容还是其中发生变化的内容（引自 5.2 节）。[F3.6（Windows）、S4、IE8]

A.6　多媒体

- <canvas> [<canvas><p>Alternative content</p></canvas>]——支持通过 JavaScript 创建基于矢量的图形（引自第 6 章）。[C4、F3、IE9、S3.2、O10.1、IOS3.2、A2]
- <audio> [<audio src="drums.mp3"></audio>]——浏览器原生播放音频(引自 7.3 节)。[C4、F3.6、IE9、S3.2、O10.1、IOS3、A2]
- <video> [<video src="tutorial.m4v"></video>]——浏览器原生播放视频（引自 7.4 节）。[C4、F3.6、IE9、S3.2、O10.5、IOS3、A2]

A.7　CSS3

下面是在 11.1 节引入的内容。

- :nth-of-type [p:nth-of-type(2n+1){color: red;}]——寻找所有满足特定类型的全部 *n* 个元素（引自 4.1 节）。[C2、F3.5、S3、IE9、O9.5、IOS]
- :first-child [p:first-child{color:blue;}]——寻找第一个子元素(引自 4.1 节)。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- :nth-child [p:nth-child(2n+1){color: red;}]——向后查找特定子元素(引自 4.1 节)。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- :last-child [p:last-child{color:blue;}]——寻找最后一个子元素(引自 4.1 节)。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- :nth-last-child [p:nth-last-child(2){color: red;}]——向前查找特定子元素（引自 4.1 节）。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- :first-of-type [p:first-of-type{color:blue;}]——寻找指定类型的第一个元素（引自 4.1 节）。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- :last-of-type [p:last-of-type{color:blue;}]——寻找指定类型的最后一个元素（引自 4.1 节）。[C2、F3.5、S3、IE9、O9.5、IOS3、A2]
- 列支持 [#content{ column-count: 2; column-gap: 20px;column-rule: 1px solid #ddccb5; }]——将内容区域分割成多列（引自 4.3 节）。[C2、F3.5、S3、O9.5、IOS3、A2]
- :after [span.weight:after { content: "lbs"; color: #bbb; }] 同 content 一起使

用，用来在特定元素后插入内容（引自 4.2 节）。[C2、F3.5、S3、IE8、O9.5、IOS3、A2]

- 媒体查询 [media="only all and (max-width: 480)"]——基于设备设置应用样式（引自 4.4 节）。[C3、F3.5、S4、IE9、O10.1、IOS3、A2]
- border-radius [border-radius: 10px;]——将元素圆角化（引自 5.1 节）。[C4、F3、IE9、S3.2、O10.5]
- RGBa 支持 [background-color: rgba(255、0、0、0.5);]——使用 RGB 颜色替换带透明度的十六进制码（引自 8.2 节）。[C4、F3.5、IE9、S3.2、O10.1]
- 阴影效果 [box-shadow: 10px 10px 5px #333;]——为元素创建阴影（引自 8.2 节）。[C3、F3.5、IE9、S3.2、O10.5]
- 旋转 [transform: rotate(7.5deg);]——旋转元素（引自 8.2 节）。[C3、F3.5、IE9、S3.2、O10.5]
- 渐变 [linear-gradient(top、#fff、#efefef);]——创建渐变图片（引自 8.2 节）。[C4、F3.5、S4]
- @font-face [@font-face { font-family: AwesomeFont;
  src: url(http://example.com/awesomeco.ttf); font-weight: bold; }]——允许通过 CSS 使用特定字体（引自 8.3 节）。[C4、F3.5、IE5+、S3.2、O10.1]

A.8 客户端存储

- localStorage——以键/值对形式存储数据，与域绑定，可跨浏览器会话保持（引自 9.1 节）。[C5、F3.5、S4、IE8、O10.5、IOS、A]
- sessionStorage——以键/值对形式存储数据，与域绑定，浏览器会话结束时清除（引自 9.1 节）。[C5、F3.5、S4、IE8、O10.5、IOS、A]
- Web SQL Databases——完整的关系数据库，支持通过事务进行表创建、插入、更新、删除和选择操作，与域绑定，可跨会话保持（引自 9.2 节）。[C5、S3.2、O10.5、IOS3.2、A2]

A.9 其他 API

- Offline Web Applications（离线 Web 应用）——为离线使用定义可缓存的文件，让应用在无因特网连接的条件下也能正常运行（引自 9.3 节）。[C4、S4、F3.5、O10.6、IOS3.2、A2]
- History（历史记录）——管理浏览器历史（引自 10.1 节）。[C5、S4、IE8、F3、O10.1、IOS3.2、A2]
- Cross-document Messaging（跨文档消息机制）——在从不同域加载内容的窗口之间进行消息传递（引自 10.2 节）。[C5、S5、F4、IOS4.1、A2]

- Web Sockets——在浏览器和服务器之间建立有状态的连接（引自 10.3 节）。[C5、S5、F4、IOS4.2]
- Geolocation——从客户端浏览器中获取经纬度（引自 10.4 节）。[C5、S5、F3.5、O10.6、IOS3.2、A2]
- Web Workers——JavaScript 的后台处理功能（引自 11.2 节）。[C3、S4、F3.5、O10.6]
- 带有 WebGL 的 3D 画布——在画布上创建 3D 对象（引自 11.4 节）。[C5、F4]
- 拖放——执行拖放交互的 API（引自 11.3 节）。[C3、S4、F3.5、IE6、A2]

附录 B

jQuery 入门

要想让我们编写的 JavaScript 兼容所有主流浏览器，而代码又能简洁明了，确实是件难事。不过现在有很多现成的库可以帮我们减轻痛苦，而 jQuery 就是其中最流行的一个。jQuery 不仅易用，而且内含的库非常多，此外也易于创建回退方案。

本附录将介绍本书中用到过的 jQuery 库中的部分内容。本附录并不能作为 jQuery 文档[1]的替代读物，也不会罗列出详细的功能和方法列表，而是抛砖引玉，引导读者入门。

B.1 加载 jQuery

可以登录到 jQuery 的 Web 站点[2]来获取 jQuery 库，并直接链接到 jQuery 脚本。不过，这里我们将在代码中链接到 Google 的服务器上来获取 jQuery：

Download jquery/simple_selection.html

```
<script type="text/javascript"
  charset="utf-8"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>
```

因为同一时间浏览器只能向服务器发送有限数量的连接请求，所以如果我们将图片和脚本分发至多个服务器，那么用户就可以更快地加载我们的页面。使用 Google 的内容交付网络有一个好处，可能其他网站也会链接到 Google 来加载 jQuery，这样的话，我们网站的访问者很可能就已经在其浏览器中缓存了 jQuery。众所周知，浏览器是根据完整的 URL 来检测文件是否已被缓存的。如果我们要将 jQuery 用于笔记本电脑或者无连续因特网连接的电脑上，那就需要链接本地已保存的 jQuery 文件了。

B.2 jQuery 基础

将 jQuery 加载到页面上之后，我们就可以使用其中的元素了。jQuery 有一个函数名为

---

① 参见 http://docs.jquery.com。

② 参见 http://www.jquery.com。

jQuery()，它是 jQuery 库的核心函数。通过这个函数，我们可以使用 CSS 选择器获取元素，并在 jQuery 对象中进行封装，以便进行操控。jQuery()函数的简写形式为$()，本书中用的就是这种写法。本附录后续的内容中，我们将称其为“jQuery 函数”。下面是其运行机制。

假设我们要获取页面中的 h1 标签，方法如下：

Download jquery/simple_selection.html

```
$("h1");
```

假设我们要找带有 important 类的所有元素，方法如下：

Download jquery/simple_selection.html

```
$(".important");
```

对比上面两种方式，可以看到唯一的区别就是我们所用的 CSS 选择器不同。jQuery 函数返回的是一个 jQuery 对象，它是一种特殊的 JavaScript 对象，其中包含匹配选择器的一系列 DOM 元素。此对象有很多有用的预定义方法，可以用来操作我们已选的元素。接下来我们详细讨论其中的一些预定义方法。

B.3 修改内容的方法

本书中使用过一些 jQuery 方法来修改 HTML 内容，下面同样进行一些此类操作。

B.3.1 hide和show

通过 hide()方法和 show()方法可以方便地隐藏和显示用户界面元素。按照如下方式，我们可以隐藏一个或多个界面元素：

Download jquery/simple_selection.html

```
$("h1").hide();
```

要将其显示出来，只要调用 show()方法即可。本书中，我们使用了 hide()方法来隐藏页面部分区域，这些区域是仅需要在禁用了 JavaScript 的情况下才显示的，比如字幕或者其他回退内容。

B.3.2 html、val和attr

我们使用 html()方法来设置和获取特定元素的内部内容。

Download jquery/methods.html

```
$("message").html("Hello World!");
```

这里，我们将 h1 标签中的内容设为了“Hello World.”。

val()方法设置和获取某个表单域的值，其用法同 html()方法一样。

通过 attr()方法，我们可以获取和设置元素的属性。

B.3.3 append、prepend和wrap

append()方法可以在已有元素之后添加子元素。假设我们有一个简单的表单以及一份空的无序列表：

Download jquery/methods.html

```
<form id="task_form">
  <label for="task">Task</label>
  <input type="text" id="task" >
  <input type="submit" value="Add">
</form>
<ul id="tasks">
</ul>
```

我们可以在提交表单的时候，通过附加新元素的方式为列表创建新元素：

Download jquery/methods.html

```
$(function(){
  $("#task_form").submit(function(event){
    event.preventDefault();
    var new_element = $("<li>" + $("#task").val() + "</li>");
    $("#tasks").append(new_element);
  });
});
```

prepend()方法同 append()方法一样都是为列表附加元素，只是插入的位置是在已有元素之前。wrap()方法将选定的元素同我们指定的 jQuery 对象表示的元素封装在一起：

Download jquery/methods.html

```
var wrapper = $("#message").wrap("<div><h2>Message</h2></div>").parent();
```

本书中，我们使用这种方式创建过一些复杂的结构。

B.3.4 css和类

我们可以使用 CSS()方法为元素定义样式，如下：

Download jquery/methods.html

```
$("label").css("color", "#f00");
```

我们可以一次只定义一种样式，不过也可以通过 JavaScript 散列为元素设定多种 CSS 规则：

Download jquery/methods.html

```
$("h1").css( {"color" : "red",
              "text-decoration" : "underline"}
           );
```

然而，其实不应将样式和脚本混到一起。在特定事件发生后，我们可以通过 jQuery 的 addClass()和 removeClass()方法来添加和删除类。然后为这些类关联样式。通过捆绑 jQuery 事件和类，我们可以在表单域获得和失去焦点的时候变换其背景色。

Download jquery/methods.html

```
$("input").focus(function(event){
  $(this).addClass("focused");
});

$("input").blur(function(event){
  $(this).removeClass("focused");
});
```

这只是一个简单的示例，而我们可以通过 CSS3 中的:focus 伪类实现与其一样的功能。只不过后者在一些浏览器中无法正常运行。

B.3.5 链

jQuery 对象上的方法返回 jQuery 对象，也就是说我们可以按照如下方式将方法链起来：

Download jquery/simple_selection.html

```
$("h2").addClass("hidden").removeClass("visible");
```

需要注意的是，不要滥用链，用多了会使代码可读性变差。

B.4 创建元素

有时，我们需要创建新的 HTML 元素，以便将其添加到文档中。可以使用 jQuery 的 jQuery() 方法来创建：

Download jquery/create_elements.html

```
var input = $("input");
```

虽然使用 document.createElement("input");也可以实现，但如果使用 jQuery 函数的话，更方便调用额外的方法。

Download jquery/create_elements.html

```
var element = $("<p>Hello World</p>");
element.css("color", "#f00").insertAfter("#header");
```

这是使用 jQuery 的链辅助快速建立和操作结构的另一个例子。

B.5 事件

通常在用户与页面交互的时候会触发事件，而使用 jQuery 可以使这种交互变得非常简单。在 jQuery 中，很多常见事件都是 jQuery 对象上的简单方法。例如，可以如下所示使页面上所有具有 popup 类的链接以新窗口形式打开：

Download jquery/popup.html

```
Line 1  var links = $("#links a");
   2    links.click(function(event){
   3      var address = $(this).attr('href');
   4      event.preventDefault();
   5      window.open(address);
   6    });
```

在 jQuery 事件处理程序中，使用 this 关键字可以访问我们所操作的元素。在第 3 行，我们为 jQuery 函数传入了 this 参数，这样可以在其上调用 attr()方法，用来快速获取链接的目标地址。

使用 preventDefault()函数可以防止原始事件被触发。这样，它就不会影响到我们正在进行的工作。

B.5.1 绑定

有些事件无法被 jQuery 直接支持，这样的话，我们需要使用 bind()方法来处理它们。例如，在实现 HTML5 规范的拖放功能的时候，就需要取消 ondragover 事件。bind()的使用方法如下：

Download jquery/bind.html

```
target = $("#droparea")
target.bind('dragover', function(event) {
  if (event.preventDefault) event.preventDefault();
  return false;
});
```

注意我们在监视的事件上去掉了 on 前缀。

B.5.2 原始事件

不论是 bind()还是 click()，只要我们使用 jQuery 事件的函数，jQuery 就会将 JavaScript 事件封装在自己的对象中，并仅复制其中的一部分属性。因此，有时候我们需要获取实际的事件，

以便访问未被复制的属性。jQuery 事件中，我们可以通过适当命名的 originalEvent 属性来访问原始事件。例如，要访问 onmessage 事件的 data 属性，方式如下：

```
$(window).bind("message",function(event){
  var message_data = event.originalEvent.data;
});
```

我们可以通过这种技术访问所有原始事件的属性或方法。

B.6 document.ready

所谓“不唐突的 JavaScript”，指的是完全独立于内容的 JavaScript。与原来在 HTML 元素上添加 onclick 属性不同，此时的做法是像 B.5 节那样使用事件处理程序。在不唐突的情况下，我们为文档添加了动作，却未修改文档本身。这样我们的 HTML 便无需依赖于用户必须启用 JavaScript 了。

这个方法有一个问题，即在未提前声明的情况下，JavaScript “看”不见文档中的任何元素。可以在页面下方通过 script 标签将 JavaScript 代码括起来，但这样不能实现跨页面复用。

我们可以将代码封装到 JavaScript 的 window.onLoad()事件处理程序中，但是所有的内容完全加载完毕后才会触发此事件，因此会有延迟。也就是说，用户可能在事件触发前正在进行某些交互。我们需要一种方法，在 DOM 加载完毕但尚未显示的情况下添加事件。

jQuery 的 document.ready 函数就是这样的作用，可以跨浏览器运转，使用方法如下：

Download jquery/ready.html

```
$(document).ready(function() {
  alert("Hi! L am a popup that displays when the page loads");
});
```

有一种简写方式是我们将会在代码中全面采用的，如下面的代码所示：

Download jquery/ready.html

```
$(function() {
  alert("Hi! L am a popup that displays when the page loads");
});
```

本书中几乎所有的示例都使用了这种模式，这样，我们便能轻松地、不唐突地为项目添加回退方案。

这里讲解的只是 jQuery 功能的一小部分。除了文档操作功能之外，jQuery 还提供了序列化表单和建立 Ajax 请求的功能，还包括一些实用函数用于轻松地循环和遍历 DOM。在我们熟悉了其用法之后，无疑会在自己的项目中发掘其更多的用处。

附录 C

音频和视频编码

对音频和视频进行编码，使其用于 HTML5 的 audio 和 video 标签中，是个复杂的课题，超出了本书的讲述范围。不过如果开发人员需要自行准备一些内容的话，通过本附录的内容可以找到一个正确的方向。

C.1 音频编码

为了适应广泛的听众，我们需要同时准备 MP3 和 Vorbis 两种格式的音频文件。这其中要用到几个工具。

在 MP3 文件编码方面，Lame 可以提供最好的音质。在编码的时候推荐用动态比特率。通过下述方式可以得到高质量的编码：

```
lame in.wav out.mp3 -V2 --vbr-new -q0 --lowpass 19.7
```

对于 Vorbis 音频来说，需要使用 Oggenc 来编码。要用动态比特率编码好音质的 Vorbis，操作如下：

```
oggenc -q 3 inputfile.wav
```

在 Hydrogen Audio[①]中可以找到关于 MP3 和 Vorbis 编码方面的更多内容。那里的信息非常棒，但是我们需要对设置多做一些试验，对自己负责也对听众负责。

C.2 为 Web 进行视频编码

在使用 HTML5 视频的时候，为了使其兼容多种平台，我们需要将视频文件编码为多种格式。不论是使用 FFMpeg[②]之类的开源编码器还是实时编码，H.264、Theora 和 VP8 都比较耗时。对视

---

① Lame 在 http://wiki.hydrogenaudio.org/index.php?title=Lame#Quick-start-28short-answer.29，Vorbis 在 http://wiki.hydrogenaudio.org/index.php?title=Recommended_Ogg_Vorbis.o。

② 参见 http://www.ffmpeg.org/。

频进行正确编码已经超出了本书的讲述范围。对于使用 WebM 容器将文件转换为 VP8 来说，我们并没有足够的版面去介绍这些命令。

```
ffmpeg -i blur.mov
       -f webm -vcodec libvpx_vp8 -acodec libvorbis
       -ab 160000 -sameq
       blur.webm
```

若不想进行麻烦的设置，可以使用 Zencoder[①]Web 服务。Zencoder 可以将视频编码为 HTML5 所需的各种格式。将视频放在 Amazon S3 或者其他公共 URL 中，然后就可以通过其 Web 界面或者调用 API 来设置多种格式的转换任务。Zencoder 会获取视频文件，进行编码，然后将新生成的视频发回我们的服务器。虽然此服务不是免费的，但其效果很好，而且如果我们要编码的内容很多的话，它还能帮助节省大量的时间。[②]

如果开发人员想自行转换这些格式的话，Miro Vider Converter[③]是另一个不错的选择。该软件预置了多种输出格式，而且是开源的。

① 参见 http://www.zencoder.com/。

② 我认识 Zencoder 的一些开发人员，不过即便不认识，也会推荐这个服务的。

③ 参见 http://mirovideo converter.com/。

附录 D

资　　源

Web 上的资源

Apple—HTML5 .... http://www.apple.com/html5/

Apple 关于 HTML5 和 Web 标准的页面，与其 Safari 5 Web 浏览器同步。

CSS3.Info http://www.css3.info/

与 CSS3 各种模块相关的很多背景信息和示例代码。

Font Squirrel .... http://www.fontsquirrel.com

提供多种免版税的字体，可在 Web 上分发。

HTML5 .... http://www.w3.org/TR/html5/

W3C 上真正的 HTML 规范。

HTML5—Mozilla Developer Center https://developer.mozilla.org/en/html/html5

HTML5 的 Mozilla 开发者中心页面。

Implementing Web Socket Servers with Node.js http://www.web2media.net/laktek/2010/ 05/04/implementing-web-socket-servers-with-node-js/

介绍如何使用 Node.js 编写 Web Sockets 服务器。

Microsoft IE9 Test-Drive .... http://ie.microsoft.com/testdrive/

在 Internet Explorer 9 中关于 HTML5（及其相关）功能的演示。

Ruby and WebSockets—TCP for the Browser .... http://www.igvita.com/2009/12/22/ruby- websockets-tcp-for-the-browser/

关于 em-websocket 的信息，是一款用于搭建 Web Sockets 服务器的 Ruby 库。

Setting Up a Flash Policy File .... http://www.lightsphere.com/dev/articles/flash_socket_policy.html

包含对 Flash 套接字策略文件的详细描述。

Typekit .... http://www.typekit.com

一款服务，让开发人员在其网站上能通过简单的 JavaScript API 使用得到许可的字体。

Unit Interactive: “Better CSS Font Stacks”.............. http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks/

对字体栈的讨论，包含一些非常棒的例子。

Video for Everybody! .... http://camendesign.com/code/video-for-everybody

关于 HTML5 视频的信息，包含让视频兼容所有浏览器的代码。

Video.js .... http://videojs.com

辅助播放 HTML5 视频的 JavaScript 库。

When Can I Use.. .... http://caniuse.com/

对于 HTML5、CSS3 和相关技术的浏览器兼容性列表。

附录 E

参考书目

[Hog09] Brian P. Hogan. *Web Design For Developers*. The Pragmatic Programmers, LLC, Raleigh, NC, and Dallas, TX, 2009.

[HT00] Andrew Hunt and David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley, Reading,MA, 2000.

[Zel09] Jeffrey Zeldman. *Designing With Web Standards*. New Riders Press, New York, third edition, 2009.

HTML5 and CSS3

Develop with Tomorrow's Standards Today

HTML5和CSS3实例教程

HTML5和CSS3代表着Web开发的未来，虽然相关规范还未最终敲定，但最新版浏览器和移动设备都已支持HTML5和CSS3。本书将带你领略现今可用的HTML5元素和CSS3特性，并提供了对旧浏览器的向下兼容解决方案，使开发人员避免因此丢失用户。

如果你还在为给按钮添加不同样式而大量添加标记，不妨拿起本书，学习一下HTML5和CSS3新特性吧。HTML5新标记可以呈现更好的结构和表单界面，编写出更为整洁易读的代码。

如果不想使用Flash，不妨看看本书是如何在页面中嵌入音频、视频和矢量图的。此外，书中关于Web Sockets、客户端存储、离线缓存和跨文档消息机制的内容将为你免去不少Web开发之苦。简单的CSS3亦将丰富页面区域的样式。如果你作为Web设计师担心旧浏览器的兼容问题，本书中相应的解决方案将为你排忧解难。

未来已在眼前，前进吧！

- Web和移动开发必读
- 掌握技术走向，自信应对未来
- 轻松实用、细致入微

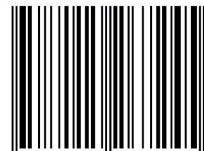
The Pragmatic Programmers

图灵社区：www.ituring.com.cn
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议 计算机/Web开发

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-26724-5



ISBN 978-7-115-26724-5

定价：39.00元

欢迎加入

图灵社区

最前沿的IT类电子书发售平台

电子出版的时代已经来临。在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。作为国内第一家发售电子图书的IT类出版商，图灵社区目前为读者提供两种DRM-free的阅读体验：在线阅读和PDF。

相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子书出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

最方便的开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版和开源出版的梦想。利用“合集”功能，你就能联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者。（收费形式须经过图灵社区立项评审。）这极大地降低了出版的门槛。只要你有写作的意愿，图灵社区就能帮助你实现这个梦想。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果你有意翻译哪本图书，欢迎你来社区申请。只要你通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

最直接的读者交流平台

在图灵社区，你可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。

你可以积极参与社区经常开展的访谈、审读、评选等多种活动，赢取积分和银子，积累个人声望。